NISTIR 5295

Guide to Software Engineering Environment Assessment and Evaluation

B.B. Cuthill

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Gaithersburg, MD 20899

NIST

NISTIR 5295

Guide to SoftwareEngineering Environment Assessment and Evaluation

B.B. Cuthill

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Gaithersburg, MD 20899

November 1993

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arati Prabhakar, Director

Guide to Software Engineering Environment Assessment and Evaluation

Barbara Cuthill

June, 1993

Table of Contents

Executive Summary

This guide outlines general approaches to software engineering environment assessment and evaluation. This guide defines SEE assessment as the process of accurately describing the capabilities of a software engineering environment to support and integrate a range of CASE tools. The guide defines software engineering environment evaluation as the process of determining how well a SEE matches the customer's requirements. Thus, the evaluation process builds on the assessment process by using the descriptions of software engineering environments capabilities generated by the assessment process to evaluate the software engineering environment with respect to specific customer requirements or criteria. While there are several models of tool assessment, evaluation and selection [e.g., 11, 14, 21, 23], there has been very little work done in assessing or evaluating full software engineering environments or environment frameworks, in part because these are new products [14].

The assessment and evaluation approaches presented here focus on accurately defining the integration and functional capabilities of candidate software engineering environments and the requirements of the customer. The NIST/ECMA Framework for Software Engineering Environments [24] and the NGCR PSESWG Reference Model for Project Support Environments [25] provide the definitions of software engineering environment capabilities used in these approaches. Wasserman [28], Thomas and Nejmeh [27] and Brown [3] provide the definitions of integration capacity used to describe the extent of integration present in candidate software engineering environments. The results of the assessment process are accurate descriptions of the capabilities of selected software engineering environments given in terms of accepted definitions or models. The results of the evaluation process are descriptions of how well candidate software engineering environments match the requirements with respect to the needs of the customer organization and compared to other software engineering environments.

Section 1. Introduction

A software engineering environment (SEE) is an interconnected collection of computer aided software engineering (CASE) tools providing automated support for software engineering development activities [14]. It includes tools supporting part or all of the software development process across portions of the software lifecycle. It may also include tools supporting the management of the software development process.

The purpose of *assessing* a SEE is to describe accurately with reference to an accepted model what the capabilities of a SEE are to support and integrate a range of CASE tools and to support specific software development activities. The purpose of *evaluating* SEEs or SEE components is to determine how well the SEE meets a set of customer requirements. Assessments are useful to the evaluation process because they provide a basis for comparing SEEs to each other or to some standard criteria.

This guide outlines general approaches to SEE assessment and evaluation focusing on accurately defining the integration and functional capabilities of candidate SEEs and the requirements of the customer. The evaluation process builds on the assessment process by using the descriptions of SEE capabilities generated by the assessment process to evaluate the SEE with respect to specific customer requirements defined in terms of the same model. While there are several models of tool assessment, evaluation and selection [eg., 11, 14, 21, 23], there has been very little work done in assessing or evaluating full software engineering environments or environment frameworks, in part because these are new products [14].

The ability to accurately assess and evaluate SEEs for a particular purpose is important because SEEs are expensive investments with potentially high risk for software developers. The software developer must find a SEE that will meet his current needs. A SEE must also evolve over time to match the changes in the developer's process and the best tools for upgrading the SEE may not always come from the same vendors. CASE tools and SEEs have until now not been able to meet this challenge in part because the capabilities of these products have not matched the needs of software developers. With most CASE tools abandoned within 6 months of purchase [8, 16, 18], the importance of matching the more complex SEEs to individual and organizational requirements cannot be overstated [26]. SEE assessment and evaluation techniques supporting this matching process will help software developers more accurately select SEEs in the future.

The rest of this paper will define SEE capabilities, assessment methods for those capabilities and evaluation methods for SEEs.

Section 2: Background: Software Engineering Environments

2.1 Software Development Activities

A SEE is a complex software product supporting part or all of an organization's software development activities across portions of the software life cycle. A software development organization performs a number of activities which a SEE can support using processes and methodologies which a SEE can reinforce. To assess and evaluate SEEs, the evaluator must define those software development activities the customer organization expects the SEE to support. A partial list of the types of activities a SEE may support follows:

a) Project Management
b) Process Control
c) Data Collection on the Software Development Process
d) Requirements Collection and Analysis
e) Design
f) Code Generation and Documentation
g) Test, Verification and Validation
h) Migration and Maintenance of Final Product

Better management of projects and processes is important to the growth of a software development organization's abilities as an organization. For example, for an organization to meet the level 3 requirements and higher of the Software Engineering Institute (SEI) Capability Maturity Model (CMM) [20], the organization must begin collecting information on its projects and using that information for planning, scheduling and process improvement. These tests fall within project management and process control activities as listed above. An organization must also begin developing a software engineering process, using that process in its projects and collecting data about that process. Various other tasks supporting process evolution and improvement also fall within the scope of project management and process control such as quality assurance.

The remaining software development activities listed relate to the technical activities of the software development life cycle. While specific software engineering methodologies interleave these activities differently, the basic activities are similar for most methodologies. These are the collection and analysis of requirements; the generation of an initial design with gradual refinements; the generation and refinement of code with accompanying documentation; the test, verification and validation of that code and the maintenance of the code with accompanying documentation. Clearly, the product of each activity is useful in pursuing the rest of the life cycle and should be available to later activities in the software development process. A good SEE can monitor the software development projects and make these products available as needed.

2.2 Development of Software Engineering Environments

SEEs are the result of the growth of software projects over the last 25 years, the need to relate products of different stages of the software development life cycle, the growing number of software development artifacts (code libraries, etc.) available for reuse and the proliferation of CASE tools. Software developers and engineers initially began using CASE tools to automate some software development activities. As software projects have continued to grow, and the use of CASE tools for many phases of the software life cycle has proliferated, software developers have a growing need for integrated CASE tools combined into SEEs. Because SEEs could support management of software engineering processes and all phases of the software engineering lifecycle, SEE technology could improve the productivity, efficiency and quality of software development.

SEE technology may prove especially useful in controlling the management of software development. In parallel with the growing interest in CASE tools, there is a growing awareness that the software industry needs better methods of managing the software engineering process [14]. Because a SEE can incorporate tools from different portions of the software lifecycle as well as management tools, a SEE can enforce or reinforce the use of a software methodology. SEE technology can reinforce a methodology by incorporating management tools to collect data on the software development process as it proceeds. SEEs can also coordinate the software development process by configuring the environment and enforcing the use of specific policies. If management tools are part of the engineering environment, they become much easier to use. Using tools to better manage the software development process can potentially lead to process improvement, better management and quality control.

To meet these long term objectives for SEE use, future CASE tools and SEEs must be interoperable. This software change reflects the hardware changes organizations have undergone in shifting from single source mainframe environments to networks of workstations from multiple vendors. The best workstation for a particular job or available at a particular time is often different from others currently in use. For similar reasons, the best software for a project or available at a specific time may not always come from the same vendors. This proliferation of software and hardware platforms has fueled the push for interoperable software especially CASE tools, SEEs and other software representing a substantial investment in capital and training time. CASE tools and SEEs must adapt to a wide range of software and hardware platforms and work with other vendors' software in order to compete in the evolving marketplace.

One means of supporting interoperability is the use of standard framework architectures. SEE frameworks include the supporting capabilities available to all CASE tools running in a particular environment. Primarily framework capabilities support inter-tool communication, integrated operation and tool-user communication. If a tool developer can count on the availability of capabilities from the SEE framework, the developer does not have to duplicate those services in the tool. Standard framework architectures also isolate those parts of the SEE which depend on specific hardware and software. If the CASE tools use only framework capabilities or services as a platform, then only the framework software must be interoperable.

However, SEEs are not a panacea. Each software development organization must learn to evaluate its own development needs and the SEEs currently on the market to find the best fit between product and organization[17, 26]. The purpose of evaluating SEEs is to establish how well different SEEs meet the customer organization's needs. An organization's needs include providing support for both existing software development processes and long term organizational goals for evolving processes. Therefore, a SEE assessment should include establishing what the current capabilities of an individual SEE are and how difficult the SEE is to expand or upgrade to meet expected future needs of the organization.

SEE technology can provide benefits to a software development organization only if the technology reflects the needs of that organization. While CASE tools have received a great deal of publicity, most organizations purchasing CASE tools do not use the tool after 6 months [16, 18]. This is largely due to a mismatch between tool functionality and organizational need. One reason for this mismatch is that few software development organizations use a consistent software development methodology. An organization should have a defined software development methodology in place before seeking tool support for that methodology [16, 21, 26]. For an organization to successfully use a SEE, it is even more important that the organization have a software development methodology before choosing the components to form a SEE supporting it. For a software development organization to successfully integrate a CASE tool into its operation, management must commit to supporting both the CASE tool or SEE and its software development methodology. For a SEE to be useful, it must match the organization's needs since a fully populated SEE can potentially play a role in all phases of the organization's software development process and the management of that process [26].

While there are many CASE tools on the market, a software development organization will find few good tools that support any particular software development process, and there are very few full software engineering environments or SEE frameworks. This SEE shortage is due to the relatively new nature of the area and the long lead time required for the development of complex software products. Also, standards for such products are rapidly evolving changing the requirements for SEE products which further lengthens development time.

Section 3: Software Engineering Environment Characteristics

The important characteristics for assessing and evaluating a SEE are those affecting its immediate usefulness to the software developer and its capacity for expansion and upgrading in the long term. The usefulness of the SEE to the developer depends on whether the SEE provides the functionality the customer needs, how that functionality is integrated, how tailorable the SEE is to the customers needs and how well the SEE performs. The long term capacity of a SEE for expansion and upgrading depends on the structure of the SEE and its adherence to the use of standard interfaces which SEE vendors will continue to use in the future.

Therefore, the significant characteristics for the assessment and evaluation of SEEs are the functionality, integration mechanisms, and structure of the SEE. The functionality of the SEE refers to the range of current capabilities available for the SEE and the future expected improvements and additions to the SEE. The SEE's integration mechanisms determine how easily the tools making up the SEE can work together and how easy it is to add new tools to the SEE while maintaining the same ease of use. The structure of the SEE refers to how separable the SEE framework providing the integration mechanisms is from the tools providing the functionality of the SEE. If the framework and tools are separate, the tools are easier to adapt, change or upgrade without affecting other parts of the SEE. How well the SEE performs its intended functions will depend on the structure of the SEE, its integration mechanisms and what kind of functionality it provides.

3.1 SEE Structure

Because a SEE is a complex software package providing a variety of capabilities in various combinations, there is no one required architecture for a SEE. However, there are implications for the integration, performance, interoperability and modification of a SEE depending on how separable the SEE *framework* capabilities are from the rest of the SEE. The SEE framework consists of "the [relatively] fixed infrastructure capabilities which provide support for processes, objects, or user interfaces" [24] as defined in the National Institute of Standards and Technology (NIST)/European Computer Manufacturers Association (ECMA) Reference Model for Frameworks of Software Engineering Environments (NIST/ECMA Framework Model) [24]. The SEE's structure consists of these framework capabilities, their relationship to the rest of the SEE, and the interface standards used to access them.

3.1.1 Frameworks: background and definitions

The SEE framework encapsulates the tools providing the capabilities for the tools to communicate with each other and with the user. These communication capabilities are essential for tool integration and process management. The framework can provide key integration capabilities for the SEE allowing for the establishment and automatic enforcement of desired relationships among the tools and software engineering activities. The framework can provide this support through a repository for the software engineering process products, a set of standard interfaces for using that repository, tools relating to the administration of the framework, and

tools for controlling and reinforcing management polices. A SEE can enforce a methodology through connections between tools requiring developers to complete or test certain software development products before using the next tool on those products. The SEE's repository could support requirements on the connections between tools and could automatically derive data about the product of one tool for use in another. The SEE framework can potentially provide the capabilities to support platform, process, presentation, data and control integration. The key framework elements supporting tool integration and process management in a SEE are a repository, a consistent user interface, inter-tool communication and SEE configuration administration support.

A key framework element is the use of a repository with standard interface support and agreed schemas for data generated and stored by all the tools in the SEE. Such a repository can store the products of all the tools making them available to other SEE tools as necessary. This retrieval process can support the software development process by reformatting the product of one tool for the use of another, deriving information about the product of one tool for use in another, automatically sending messages to tools about the status of a project and providing a standard place to find particular types of data. Storage and retrieval of software development artifacts can also include making those artifacts available for reuse on future projects. Similarly, future projects can make use of automatically collected management or process information for comparing projects.

Another key framework element is a user interface package with a standard interface and conventions for screen layout and user interaction. A similar user interface across all the tools reflecting an agreed model of user interaction makes using the tools much easier. The user can apply knowledge gained using one tool to the other tools. Similarity of user interfaces can also make the SEE appear to move from one tool to another easily.

A third key framework element is the inter-tool communications mechanism with standard conventions for types of messages tools can send, actions tools can take in response to messages or other types of communication between tools. The communications mechanism can support the integration of new tools. For example, the SEE can restrict intertool communication to messages passed between tools or support broadcast messages available to any tool. Whatever the mechanism for sending messages, the messages actually sent have to meet conventions that the tools can process and act on.

A fourth key framework element is the use of tools supporting framework and configuration administration for the SEE and for determining, controlling, measuring and enforcing the use of software engineering processes. By providing for framework and configuration administration to tailor a SEE to a given set of requirements, the SEE user can create the best environment for his software development process. Similarly, by providing for control and enforcement of software engineering processes, managers can gain control of project planning and scheduling and have a better understanding of what processes work in a given organization.

3.1.2 *NIST/ECMA Reference Model for Frameworks of SEEs*

The *NIST/ECMA Reference Model for Frameworks of SEEs* [24] (NIST/ECMA Framework Model) provides a comprehensive description of SEE framework capabilities. The NIST/ECMA Framework Model is "a conceptual (and functional) basis for describing and comparing" SEE frameworks [24]. The NIST/ECMA Framework Model distinguishes between the tools supporting development tasks and the framework encapsulating the tools and supporting the developers use of those tools. The purpose of the NIST/ECMA Framework Model is *not* to describe the capabilities of a full software engineering environment. The functionality of the software development tools is not part of the model. Thus, the NIST/ECMA Framework Model assumes a SEE consists of a framework surrounding software development tools supporting the software engineering process.

Edition 3 of the NIST/ECMA Framework Model contains descriptions of functions or capabilities termed *services*. There are 66 services grouped into 7 major service groups. The major service groups contain services which support the use of a common resource, support a particular user role or have some other feature in common. The definitions of these services and groups are taken from the final draft of Edition 3 of the NIST/ECMA Model which NIST and ECMA will publish in 1993. Sections A.4 and A.5 contain the definitions of all the NIST/ECMA Framework Services. The major service group definitions follow:

Object Management Services - The general purpose of the object management services are the definition, storage, maintenance, management, and accessing of object entities and the relationships among them in a repository.

Process Management Services - The general purposes of the Process Management Services in a SEE are the unambiguous definition and the computer-assisted performance of software development activities across total software life cycles. In addition to technical development activities, these potentially include management, documentation, evaluation, assessment, policy enforcement, business control, maintenance, and other activities.

Communication Services - This service provides a standard communication mechanism which may be used for inter-tool and inter-service communication. The services depend upon the form of communication mechanism provided: messages, process invocation and remote procedure call, or data sharing.

User Interface Services - The user interface services define a standard user interface all tools can access for user-tool communication with a consistent look and feel across the entire SEE.

Operating System Services - The set of underlying operating system and platform services supporting the framework services.

Policy Enforcement Services - The reference model uses the term "policy enforcement" to cover the similar functionality of security enforcement, integrity monitoring, and various object

management functions such as configuration management. The framework reference model regards security as a service crossing many of the boundaries of the reference model's major service groups.

Framework Administration and Configuration Services - A SEE framework has to be carefully administered because its precise configuration may be constantly changing to meet the changing needs of the software development enterprise. These services support the framework administrator's activities in controlling the SEE.

The NIST/ECMA Framework Model is a comprehensive guide to the services that a SEE framework *can* provide. Every SEE framework does not have to include every service. Particular user communities or tools may not need every service. However, the failure to provide a framework service has implications for the types of tools, tool functionality, and/or tool integration that the SEE will support and may limit the capacity of the SEE in the long term.

3.1.3 Implications of framework choices

The structure of the SEE determines whether the user can add new tools to the framework easily, which tools the user can add and the extent of the integration possible among the tools. The SEE structure consists of the relationship of the framework capabilities to the rest of the SEE and the interface standards used to access the framework. Using a framework separable from the tools allows for the addition of tools to the system, with minimal system disruption, as new tools or upgraded tools appear in the marketplace. If the framework capabilities are not separate from the tool functions, the user will find it difficult to add, replace or upgrade tools. The changed or added tool must provide the framework services needed to integrate it with the existing tools instead of relying on the provided framework services.

Similarly, using standard, non-proprietary interfaces to the framework capabilities can enhance the range of tool choices in the long term. Proprietary interfaces to a SEE limit its future extension or evolution to products available from the specific supplier(s) owning or licensing that interface. If those software vendors leave the market, the customer cannot upgrade his system. Choosing products meeting an open standard allows the customer greater flexibility because any vendor can make tools to that standard. Because no one standard has achieved industry dominance, the customer must consider the competing standards carefully since the standard chosen determines what tools he can add to the SEE in the future.

3.2 SEE Functionality

Because a SEE is a complex software package providing a variety of capabilities in various combinations, there is no one set of functions a SEE must provide. Different software development organizations should tailor their SEEs with different levels and types of support for different phases of their software engineering process. The maturity of the organizations' process and the type of software under development are just some of the criteria influencing the software development organization's tool selections.

3.2.1 Populated frameworks: background and definitions

Populated frameworks are SEE frameworks with tools added to support at least some of an organization's software development activities. A full SEE supports the entire software development process including all the software development activities and the management of those activities. There are no full SEEs at the present time and most software development organizations do not need and do not have a mature software development process that could use the functionality of a full SEE.

Populated frameworks contain the tailored set of tools an organization needs to perform its software development process. In order to populate a framework with useful tools, the software development organization must be able to assess its own software development process and activities well enough to define the support these activities can successfully use [26]. These tools provide the functionality of the SEE.

3.2.2 NGCR PSESWG Model of Project Support Environments

A comprehensive listing of SEE functions is provided in the Next Generation Computer Resources Program (NGCR) Project Support Environment Standards Working Group (PSESWG) Reference Model for Project Support Environments (PSE) (PSESWG Model) [25]. The US Navy's NGCR Program is intended to fulfill the US Navy's need for standard computer resources through the selection of interface standards. NGCR established the PSESWG to examine interface standardization within project support environments. PSESWG's long term goal is the identification of a profile of industry accepted interface standards for PSEs that the Navy can use. These standards will reflect the Navy's need to support a wide range of project development activities within a single environment and will guide Navy procurement of PSEs.

To further its goal of selecting compatible interface standards, PSESWG developed a reference model for PSEs. PSEs are environments supporting the engineering, development and maintenance of computer-based systems. These systems could contain a mix of hardware and software components. Most definitions of SEEs limit SEE functions to those supporting software development and processes in the software lifecycle. Another difference between the PSESWG Model and most SEEs is that the PSESWG Model describes a PSE for *mission critical* systems; therefore, the PSESWG Model includes capabilities not necessary for the development of most software. While the mission of NGCR PSESWG was to support Navy activities, its work benefits all SEE customers by providing a standard reference model.

The PSESWG Model includes the functions of a full SEE. It is "a conceptual description of the functionality provided by a [PSE] ... bounded neither by a particular application domain or lifecycle paradigm." [p. 5, 25] Like the NIST/ECMA Framework Model, which PSESWG incorporates, this model defines capabilities for describing and comparing SEEs and SEE components. The two models take the same approach to describing the capabilities associated with components of SEEs differing primarily in scope. The NIST/ECMA Framework Model describes SEE frameworks while PSESWG describes full SEEs putting the NIST/ECMA

Framework Model into a broader context.

The PSESWG Model describes the functions or capabilities of a PSE as services in a similar manner to the NIST/ECMA Framework Model. Both reference models contain services clustered in service groups. The primary difference between the service descriptions in the PSESWG and the NIST/ECMA Models is in the granularity of those descriptions. The NIST/ECMA Framework Model is more fine-grained than the PSESWG Model. The NIST/ECMA Framework Model typically describes a major SEE component or tool using a service group. The PSESWG Model typically describes a major SEE component or tool with a single service while PSESWG service groups typically correspond to related collections of tools or toolsets.

The PSESWG Model divides a PSE into 5 major service groups related to the role of the user of those services and the function of the services. These service groups follow:

- Technical Engineering Services - used by project developers
- Technical Management Services - used by senior project personnel/project leaders
- Project Management Services - used by project managers/planners
- Support Services - for all users
- Framework Services - used by system administrators and by the tools

The definitions of the 5 major service groups and the sub-groups contained in each follow. These definitions paraphrase the ones in the NGCR PSESWG Reference Model for Project Support Environments, Version 1 [25]. Appendix A contains descriptions of all the PSESWG Model services.

The **Technical Engineering Services** support the actual specification, design, implementation and maintenance of systems. This service group contains three major sub-groupings: system engineering services, software engineering services and life-cycle process engineering services. The system engineering services includes the tool capabilities used to develop systems including substantial hardware as well as software components. The software engineering services include the capabilities associated with most current CASE (Computer Aided Software Engineering) tools. The life-cycle process engineering services are those capabilities supporting the definition and use of good software development processes. While tools providing process support do not yet exist, the goal of including the life-cycle process support services is to show the need for these tools and to show how they would be part of future PSEs or SEEs. Descriptions of all the services defined as part of the technical management services are given in Section A.1.

The **Technical Management Services** support the senior project personnel in collecting data necessary for the management of the product. There are 4 services within this service group: the configuration management service, the change management service, the reuse management service, and the metrics service. The configuration management service identifies, documents and controls the functional and physical characteristics of the work product of the PSE to ensure the

traceability and reproducability of the product. The change management service supports the creation, evaluation and tracking of change requests used to modify a work product of the PSE under configuration control. The reuse management service supports the storage, examination and reuse of project assets such as components of the requirements, design, code or test cases used in a previous project. The capabilities of current tools in the reuse area are very limited but there is a great deal of research on this subject. The metrics service supports the collection and analysis of data about the work product of the PSE. Section A.2 contains a more detailed description of these services.

The **Project Management Services** support management by providing the capabilities necessary for the management of the resources used in generating the work product of the PSE. The project management services consists of 4 services; the scheduling service, the estimating service, the risk analysis service and the tracking service. The scheduling service provides the capabilities needed to schedule the resources and tasks used to generate the project. The scheduling service supports the application of chronological constraints to a project. The estimating service supports qualification, analysis and prediction of project cost and resource needs. The estimating service supports the application of resource constraints to a project. The risk analysis service supports the comparison and analysis of actual data about the development of the project with projected values and of actual data about the work product with expected values. The tracing service supports the correlation of estimated cost and schedule data with actual project performance. Section A.3 contains a more detailed description of these services.

The **Support Services** are those services providing capabilities that all the users need at some time. This service group contains 5 major services or service groups: the common support services, the publishing service, the presentation preparation service, the user communication services, and the administration services. The common support services define the capabilities for manipulating standard representations of various types of data such as textual or numeric data. The publishing service provides the capability to create and print documents. The presentation preparation service provides the capability to create materials for presentations. The user communication services support interaction between PSE users. The administrative services provide the capabilities for the user to interact with the PSE as a whole, to configure the PSE, and to tailor the PSE to specific needs. The complete list of definitions for the services contained in the support services group are given in Section A.4.

The **Framework Services** support the tools populating the PSE and provide the infrastructure connecting those tools to each other and to the user. The framework provides the integration mechanisms for the PSE. The definitions for the framework services are taken from the NIST/ECMA Reference Model for Frameworks of Software Engineering Environments [24] and the Draft Guide to the POSIX Open Systems Environment [10]. The framework services group contains 8 service groups: operating system services, object management services, policy enforcement services, communication services, user interface services, user command interface services, and network services. The operating system services define those capabilities typically associated with an operating system. The object management services define those capabilities associated with the definition, storage, maintenance, management, and access of objects. For a

PSE, these objects include the components of the work product or information about the work product. Policy enforcement services support the application of security and integrity constraints to objects. The process management services support the definition and performance of product development activities. The communication services define the inter-tool or inter-service communication capabilities. The user interface services define a set of capabilities for tools to use for communicating with the user and can provide presentation integration. The user command interface services provide a common interpreter for a user to interact with an environment. The network services provide for the transfer of information within a distributed environment. The definitions for the services contained in these subgroups are in Section A.5.

3.2.3 Constraints on functionality choices

While the purpose of the SEE is to provide increased software development support to users, the choice of frameworks, interface standards and integration standards will have a long term effect restricting the choice of functionality easily available to the SEE customer. The choice of framework limits the tools to those the framework supports. If the framework cannot provide the necessary platform for a tool, the tool is unavailable. The choice of interface and integration standards limits the tools to those compliant to the standards. If the customer wants non-compliant tools, he must obtain special purpose interfaces for the selected tools.

3.3 SEE Integration

The purpose of combining tools into SEEs is to provide the functionality that individual tools used separately do not possess such as seamless support for products across all phases of the software life cycle. Integration is the glue allowing tools and other SEE components to combine. Integrated SEEs (ISEEs) are collections of CASE tools connected by one or more integration methods. How to define, describe and measure integration in SEEs is a subject of controversy.

However, to assess SEEs, the evaluator must consider how well candidate SEE tools and frameworks interact. Two SEEs supporting identical life cycle functions but integrated to a different extent are very different SEEs. If the SEE does not provide strong integration, the user could end up wasting valuable time substituting his efforts for some of the missing integration capability. This failure could lead to abandonment of the SEE. The variety of SEE integration definitions and measurement techniques confuses the potential SEE buyer since conflicting claims about what is necessary for good SEE integration abound.

3.3.1 Integration definitions

One method of categorizing integration strategies is as *vertical* or *horizontal* integration strategies. This categorization relates to the function of the tools. Vertical integration means that the SEE integrates tools over a restricted class of functions operating over a short period of the software lifecycle. For example, the SEE may integrate tools only if they are used in the same or adjacent life cycle phases. Tools can share information only if it is necessary to complete a

specific phase or go to the next phase of the life cycle. Horizontal integration means the tools are integrated across the multiple phases of the life cycle. For example, a configuration management tool is a horizontally integrated tool if it creates versions of the product of any life cycle phase and produces consistent versions of the code and documentation making up a specific version of the product under development.

Another categorization of integration strategies is to describe them in terms of the function or purpose of integrating the tools. ISEE researchers [3, 6, 27, 28] have identified five interrelated types of tool integration based on the shared functionality or the shared product of the integrated tools. These integration types are platform integration, presentation integration, data integration, control integration, and process integration. Wasserman [28] has also defined different levels of integration describing the extent of the integration for some of the different types. An actual tool integration method or model used in an ISEE should support more than one of these types, but may provide different levels of support for different types of integration.

Platform integration is the integration mechanism allowing SEEs to interoperate. The platform integration mechanism allows all users of a SEE in a workstation environment to receive the same view of the SEE. Wasserman [28] defines platform integration as "the set of services that provide network and operating system transparency to these applications." The SEE framework should provide or supplement a good operating system providing these services insulating the CASE tools from the hardware and software platforms.

Presentation integration is the integration mechanism enforcing consistency on the user interfaces of all CASE tools in the SEE. A consistent set of user interface conventions makes use of the CASE tools easier. Platform integration supports presentation integration since using the same window system or window manager clearly overlaps with the interoperability needed for platform integration. Part of providing interoperability is providing a set of services linking the user interface requirements of the CASE tools and the windowing system available as part of the software platform.

Data integration is the integration mechanism allowing tools to share information. Tools can work together only if they can exchange information. Researchers [27] have primarily suggested three methods for data integration. These are the direct transfer method which transfers information through pipes or other mechanisms, the use of a common interface package to modify transferred information, or the use of a shared repository to indirectly share information. Sharing information also requires that tools agree on a standard format for sending and interpreting the information from the other tools. Use of a shared repository requires a set of shared schemas and conventions for locating various SEE products within the repository, information contained in the SEE products, and interpreting information in the repository.

Control integration is the process tools use to exchange and enact or execute "events." Events can be the starting or ending of particular jobs or tasks. The tool receiving the notification can then be directed to or automatically start a new task. One method of passing event information is through the use of a messaging facility. For example, the Object Management

Group (OMG) has developed a standard for the Object Request Broker (ORB) which passes control integration messages. Tools indicate to the ORB that they can receive and send specific types of messages. The ORB transfers the messages provided to it among the tools registered to send and receive them.

Process integration is the integration of the software development tools with tools supporting the software engineering process. Integrating these two types of tools requires the other types of integration as well as new integration methods. Process integration is a new area with very few products, standards for products, or definitions of what products should be. Process integration requires that tools support the user's defined process by relating tools involved in the same process step, exchanging process event information among tools and imposing process required constraints on the use of tools or the products of tools. In the future, SEE process integration mechanisms may be layered on top of control and data integration mechanisms.

Thomas and Nejmeh [27] provide an alternative way of describing types of integration. They view integration as the definition of relationships between pairs of tools. Each relationship is of one of Wasserman's types: process, presentation, control or data integration with associated properties. The "value" associated with each property is the extent to which the relationship or integration method meets the integration goal associated with that property. For example, the properties of a control integration relationship are *provision* and *use*. The provision property of a control integration relationship is the extent to which a tool's services meet the goal of being used by other tools in the environment. Similarly, the use property defines the extent to which a tool uses the services provided by other tools in the environment. The data integration properties are *interoperability* or the amount of work needed for a tool to manipulate data produced by another tool; *nonredundancy* or the amount of duplicated or derivable data managed by a tool; *data consistency* or how well the tools cooperate to maintain semantic constraints on the data; *data exchange* or how much work is required to make nonpersistent data usable by other tools; and *synchronization* or how well tools communicate changes in the data. The process integration properties are *process step* or how well tools combine to support a process step; *event* or how well tools agree on the events required by a process; and *constraint* or how well tools enforce a constraint. The presentation integration properties are *appearance and behavior* or the extent to which tools use similar screen appearances and interaction behavior and *interaction paradigm* or the similarity between tools' metaphors and models.

These varying ways of describing tool integration have an impact on SEE assessment. Integration is the key element in a SEE which transforms the SEE from a collection of related tools into a programming environment providing a range of useful functions in support of software development. To compare SEEs, the evaluator must select a method of defining and rating integration between tools and collections of tools. For the assessment method presented in this report we will rely primarily on Wasserman's definitions of types of integration [28] for describing significant integration areas and capabilities.

3.3.2 Frameworks and integration

While the NIST/ECMA Framework Model does not address integration issues directly, many of the NIST/ECMA framework services are those capabilities required to support tool integration. The NIST/ECMA Framework Model defines services supporting inter-tool, platform-tool and user-tool communication and operation. These services insulate the tools from each other and from the hardware and software platforms, allowing easy upgrading and interoperability. Integration measures define how tools should work together to support specific SEE functions. The framework services define specific capabilities while integration measures define the overall capacity of the SEE to support integration in a particular area.

The NIST/ECMA and PSESWG framework services describe key SEE elements supporting the five integration areas defined in [28]. While the framework services do not describe integration measures and the integration measures do not refer to specific framework services, it is possible to pair types of integration with the key framework services allowing a SEE to support that type of integration. One pairing of framework services and integration areas follows in Table 1.

Integration Type	**Primary Supporting Reference Model Service Groups**
Platform	Operating System Services Networking Services
Process	Process Management Services Policy Enforcement Services
Control	Communication Services
Data	Object Management Services
Presentation	User Interface Services User Command Interface Services

Table 1: Reference Model Service Groups Supporting Integration Types

While the service groups do not correspond as neatly to specific integration areas as Table 1 suggests, the integration areas are not completely separable either. A SEE framework component can easily straddle several service groups and integration areas. For example, a data repository for a SEE can support data, platform and control integration and can provide operating system, object management and process management services. However, the table does list the services most closely related to or most likely to provide most of the support for the specific integration areas independent of specific architectural considerations.

3.3.3 Defining the extent of SEE integration

The extent to which a SEE supports a specific type of integration is not an easily

measurable quantity; however, methods for classifying the extent of integration or levels of different integration types exist. Brown and McDermid [6] have proposed five levels of tool integration which are carrier, lexical, syntactic, semantic and method. An example of carrier level integration is the direct transfer method of data integration. Carrier level integration provides a file format that is readable by each tool. This is the minimal form of integration between tools. Lexical integration exists in tightly coupled tool families using a common schema among themselves. Like carrier level integration, lexical level integration requires that tools contain the information necessary to interpret any information passed to them. Syntactic level integration introduces common representations or database schema languages which all tools use. Semantic level integration introduces common schemas or conventions for providing shared meaning in inter-tool communication. Method level integration enforces process required interactions on the tools. Currently there are no products or standards supporting method level integration.

Wasserman [28] provides specific definitions of levels of integration for control, presentation and data integration. These level definitions provide area specific definitions for the first four levels of Brown and McDermid's abstract integration levels [6]. Table 2 provides Wasserman's definition of specific capabilities at different integration levels for data, control and presentation integration and how they can be related to Brown and McDermaid's abstract definitions of level integration.

Integration Type	Carrier	Lexical	Syntactic	Semantic
Data	explicit message passing	using shared files	use of the same database	use of the same object base
Control	explicit message passing	daemons	triggers	message servers
Presentation	use of the same window system	use of the same window manger	use of the same toolkit	standard semantics for the toolkit

Table 2: Definitions of Integration Levels [28]

Wasserman defines tool integration in a SEE by its position in a three-dimensional space in which each dimension defines the extent to which the SEE integrates the tools in terms of presentation, control or data.

3.3.4 SEE integration standards

A user can tailor a good SEE through his choice of tools with which to populate the SEE, but the integration or interface standards used in the SEE framework determine the future options of the buying organization. If the integration standards are open and non-proprietary, the customer has potentially a large number of vendors to choose from providing better product value and selection. Proprietary integration standards limit tool selection to tools from only one or a small consortium of vendors. Table 3 below provides the current standards available for integrating tools in a SEE. Standards in bold are non-proprietary. Currently, there are no precise

definitions of what process integration requires and what a standard for process integration would be.

Integration Type	Carrier	Lexical	Syntactic	Semantic
Presentation	X Window System	X Window Manager	X Lib	MOTIF OpenLook XVT
Data	POSIX UNIX	CAIS-A	PCTE ATIS	PCTE + Agreed Schemas
Control	Interrupts			BMS CORBA Tooltalk
Process	Not defined	Not Defined	Not Defined	Not Defined
Platform			POSIX UNIX	

Table 3: Standards/Products Supporting SEE Integration

Section 4. Software Engineering Environment Assessment Methodology

The purpose of assessing a SEE is to describe accurately with reference to an accepted model the extent of the SEE's capacity to support and integrate a range of CASE tools. Assessing SEEs requires defining SEE capabilities and the means of characterizing those capabilities. This assessment method will use the definitions for SEE framework capabilities in the NIST/ECMA Framework Model [24], the definitions for SEE capabilities in the PSESWG Model [25] and the integration definitions from [3, 6, 27, 28]. NIST, ECMA, and PSESWG have defined capabilities for SEEs and SEE frameworks in general terms. The goal of these models is a single set of defined capabilities for comparing these products and defining needed interfaces between capabilities. NIST, ECMA and PSESWG have also provided mapping guidelines [19, 24, 25] for relating SEEs to the reference models. Because no standard architecture exists for SEEs and SEE capabilities can be spread across a number of different tools in different configurations, this assessment strategy describes SEEs and SEE interfaces in terms of abstract services rather than a specific architecture or integration strategy.

While there is no reference model comparable to the NIST/ECMA or PSESWG models for SEE integration, definitions of levels and types of integration exist in [3, 6, 27, 28] which are valuable for classifying the integration found in SEEs or parts of SEEs. These classifications are useful for defining how well the customer should expect the components of the SEEs to work together. SEE integration is measurable and comparable in two ways by the type of integration mechanisms present (e.g., data, presentation, etc.) and by the extent of the integration (e.g., shared formats).

4.1 Overview of Software Engineering Environment Assessment Process

This SEE assessment process reflects the view of SEEs as frameworks populated with tools and focuses on assessing three significant characteristics of the SEE: its structure, integration capacity and functionality. It is how the structure and integration mechanisms connect the tools and framework to achieve a range of functionality that defines the usefulness and capacity for future expansion of the SEE. The SEE structure defines the potential capacity of the SEE to support tools or integration mechanisms. The SEE functionality defines what the SEE can immediately do for the user. SEE integration is the glue which connects the tools and a framework into a single SEE. Describing SEE capabilities using defined, accepted terminology is key to describing what customer organizations need, what the marketplace can supply, and what gaps exist between the two. The steps of the assessment method and the results of each step follow in Figure 1:

1) Identify candidate SEE frameworks
   Result: List of candidate frameworks and documentation describing their capabilities
2) Map framework capabilities to NIST/ECMA Framework Model
   Establish how separable the framework and tools are
   Identify any gaps in the framework
   Result: Mapping of each SEE framework to the NIST/ECMA Framework Model
   Report analyzing each mapping
3) Identify interfaces between framework and tools
   Identify standards used for those interfaces
   Identify any other integration protocols used
   Result: List of interface and integration standards and protocols that each SEE framework supports
4) Identify tools compatible to framework/integration standards
   Result: List of candidate tools compatible to each SEE framework
5) Map tool options to PSESWG Reference Model
   Establish range of tool function available
   Establish range of tools available for each function
   Result: Mapping of each SEE compatible tool and tool set to PSESWG Model
   List of tools by type available for each SEE framework
   Analysis of range of tools of each type available for each SEE

Figure 1: SEE Assessment Process

This assessment process establishes the capacity of the SEE framework to support integrated tools and the range and capabilities of the tools supported by each type of SEE framework. The rest of this section describes the assessment process in greater detail.

4.2 Identify Candidate SEE Frameworks

The SEE assessor (e.g., the individual or group performing the SEE assessment) can identify candidate environments by investigating the vendor literature, SEE surveys and other published evaluations [i.e., 14, 15], and contacting vendors directly. Obtaining information on the candidate SEEs sufficient to assess the SEE frameworks requires getting detailed information from the vendors, from other users of the SEE and, if possible, from hands-on testing of the SEEs.

The result of this step of the SEE assessment process is a list of the candidate SEE environments and documentation describing those environments' capabilities.

4.3 Map Framework Capabilities to NIST/ECMA Framework Model

4.3.1 Purpose of reference model mappings

Because one of the purposes of developing the NIST/ECMA and PSESWG reference models is to provide a mechanism for defining the capabilities of specific SEEs, the reference models are useful for comparing and assessing SEEs. By mapping actual SEEs to the reference models, it is possible to define what capabilities the SEE possesses. Figure 2a shows how mapping the SEE to the reference model can indicate which groups of services the SEE includes. Defining the capabilities of SEEs and comparing SEEs directly is difficult because they are complex software systems with varying internal structure providing a variety of different services in various configurations. The reference model allows the assessor to relate specific portions of SEEs to a set of criteria defined in terms of the service rather than a particular architecture which may not apply. Figure 2b illustrates how several SEEs can map to overlapping sets of services.

One purpose of mapping SEEs to the NIST/ECMA Framework Model services is to indicate which framework services are *not* present in a SEE. The lack of a framework service indicates possible limits on the future usefulness of the system. While this lack could be deliberate if the ability is not necessary in the SEE's intended market, a missing framework service indicates a possible limitation on the extent of integration or cooperation among SEE components or between components and the customer and may limit the future SEE expansion. It is important for the customer to realize the long term implications of missing any framework services. Alternatively, the customer may conclude that lacking a particular service is not important for his software development activities. In Figure 2a, the missing services are highlighted with an 'X'.

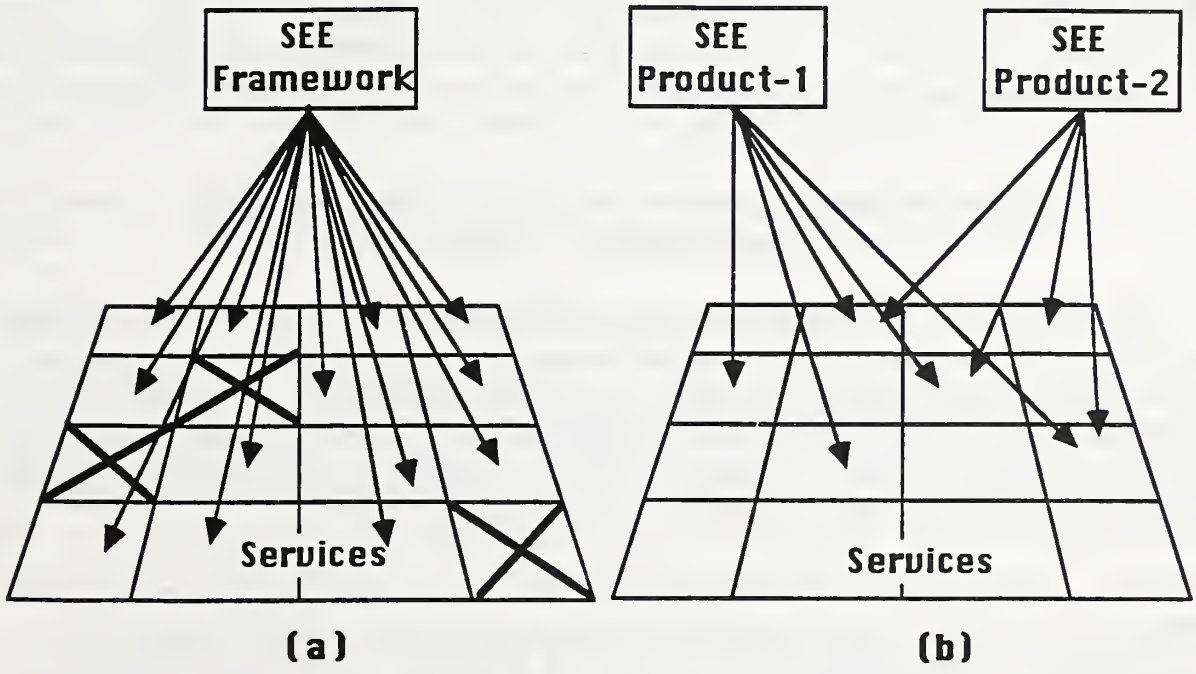

Figure 2: Mapping Illustrations
(Note: Figure adapted from those in [29])

4.3.2 Using the mapping guidelines

The mapping process requires the assessor to understand the reference model and the SEE product. The mapping process maps the services from the NIST/ECMA Framework Model to the capabilities of the target SEE product. The NIST/ECMA Framework Model is fine-grained and one reference model service may be a subset of one SEE framework component. Figure 3a illustrates this type of mapping. Alternatively, one framework service may be scattered among several SEE components. Figure 3b illustrates this type of mapping. [19] provides guidelines for mapping SEEs and SEE components to the NIST/ECMA Framework Model.

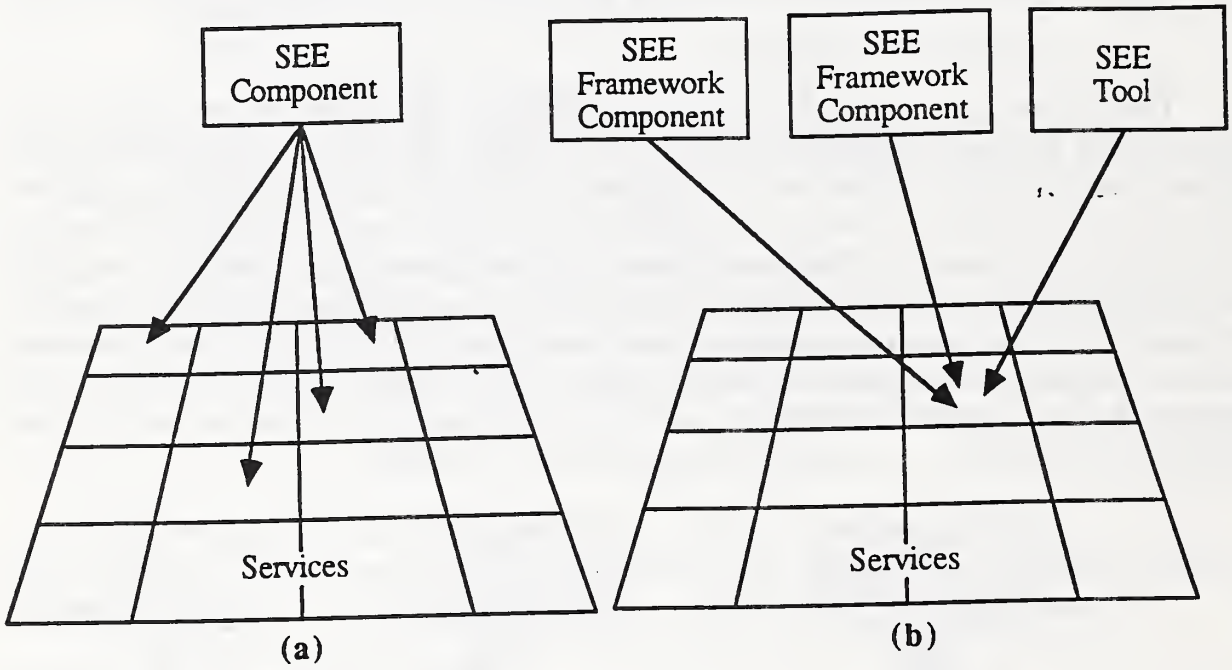

Figure 3: Correspondence of Services and SEE Components
(Note: Figure adapted from those in [29])

The NIST/ECMA Framework Model services describe complex capabilities in detail and from a variety of viewpoints (e.g. conceptual, operational). The framework model provides a structure for these descriptions by defining them in terms of *dimensions*. Each dimension defines a view of the service. Mapping SEE product capabilities to the dimensions generates more precise descriptions of the SEE product's capacity to provide the service by structuring the description of the capacity in the SEE. The definitions of the dimensions follow:

Conceptual - This dimension describes the semantics (i.e., functionality) of a service without reference to either how it is implemented or to the ways in which it may be made available to other services or to users. The reference models do not define a language or a notation for creating the conceptual description.

Operations - This dimension defines the set of operations that implement the functionality

described by the conceptual dimension. It does not include the explicit format (see external dimension) or the implementation details (see internal dimension) of that functionality.

Rules - Rules associated with the objects and operations of a service constrain the states the objects may reach and the changes to states that operations may make. Pre- or post-conditions or restrictions on the use of objects or operations are examples of rules. In addition, some services may provide the ability to define additional rules and associate them with that service. For example, a service may ensure a manager manages no more than five programmers or disallow the redefinition of access controls.

Types - This dimension describes the possible types of objects (or data model) used by that service, information about these types (metadata), as well as the objects (instances of these types) used in the service. Type information may also be found in at least three other places: objects needed by the "operations," objects needed in the "implementation," and objects provided externally for use in other services. It may be useful in some cases to make a distinction between instances, types, and information about types (metadata).

External - This dimension discusses how the service is made available for use. Other framework components, tools (or application programs), or users may use the service. For example, a query service for an information repository may provide a procedural interface, an embedded query language, or a full tool for user support.

Internal - This dimension discusses implementation issues. In general, a good design separates implementation issues from the functionality provided. Services available to tools executing within the SEE might be supplied by a specific framework implementation, by the underlying native operating system upon which the framework executes, or by other tools executing within the framework. In all of these cases, these services are considered to be supplied by the framework and the internal dimension describes the underlying implementation.

Relationship to Other Services - This dimension describes how one service might interact with another service; this may include examples of typical service interactions.

Examples - This dimension contains specific examples of the service for illustration.

To map the framework services, the assessor must go through each framework service and look for an equivalent SEE framework capability. The assessor must then restructure the description of that capability to match the dimensions. However, it is not enough to note that the SEE framework possesses the capacity associated with a service; the assessor must note what SEE component provides the framework service. The relative locations and possible recombination of SEE services are important to SEE assessment.

The assessor may encounter several types of problems in mapping SEEs to the framework model. First, the capabilities associated with a single framework service may be scattered among several SEE components as Figure 3b illustrated. This scattering is important since the user will

have difficulty updating or replacing a service which the SEE does not provide in one place. Second, CASE tools may duplicate some of the framework services. This duplication limits how well the SEE can integrate that tool and may result in problems if the framework changes. Third, a SEE component may include several very different SEE services as Figure 3a illustrated. This is a problem if the services need separate updating. Fourth, the framework may include optional components which the assessor will have to carefully identify as such and identify possible different configurations of the framework. The specific choice of tools to combine into the final SEE will change the outcome of the mapping; therefore, each tool or SEE component must be mapped to the reference models separately. These mappings can then be combined into configurations reflecting possible tool configurations. Generally, the assessor must be careful in mapping services and SEE capabilities because the two will not provide an exact correspondence. This lack of obvious correspondence requires the assessor performing the mapping to use his judgement about service/capability correspondence.

Appendix C provides more information on performing mappings drawn from [19, 24, 25].

4.3.3 Results of the mapping process

The results of mapping the NIST/ECMA Framework Model to the SEE frameworks should be a series of mappings showing

1) each possible SEE framework configuration
2) what services each configuration provides
3) which SEE components provide which SEE framework services
4) what SEE framework services are not provided in each configuration.

This information provides a good indication of how separable the SEE framework and tools are, how replaceable or upgradable components of the SEE framework are, and what limitations the framework has with respect to providing needed integration services.

4.4 Identify Interfaces Between Framework and Tools

Once the assessor has the SEE mappings, the assessor can identify the framework interfaces. These interfaces consist of the connections between the framework and the tools and should meet accepted standards. The SEE's framework capabilities determine what types of integration the SEE framework supports and the SEE's interface standards define the extent of the integration possible in each area. For example, a SEE framework may contain a data repository, but the interface to that repository, whether it is ATIS, PCTE or some new standard, defines how the tools can use the repository. The framework interfaces also define which tools are compatible to and can use the framework.

If the framework interfaces are hard to identify, the framework is tightly coupled with the CASE tools and the framework and tools are not easily separable components. If the framework and tools are not easily separable then buying tools to expand the SEE or change its

configuration is difficult or impossible. The SEE is limited to those capabilities supplied by one or a small group of vendors.

The result of this step of the assessment process is a list of the interfaces and interface standards used to connect tools to the SEE framework or to each other. If the interface standards are not publicly available, the entire SEE must be obtained from a single source and only limited variations of the SEE's capabilities may be obtainable.

4.5 Identify Tools Compatible to Framework/Integration Standards

Once the assessor has established the interfaces for the candidate SEE's framework components, the assessor can go back to the vendors and the literature and find all those tools which are available for the SEE's interface standards. As some standards become accepted, the tool vendors and SEE framework vendors will tend to write tools for those standards. SEEs designed for proprietary framework standards will always be limited to the tools available from the owners of that standard.

The result of this step of the assessment should be a list of tools available which work to specified interface standards.

4.6 Map Tool Options to PSESWG Reference Model

The assessor at this point in the process has a record of each SEE's frameworks capabilities and interface standards and can match those interface standards to lists of standards compliant tools. The assessor needs to know the functionality available for a SEE with a specific set of interface standards. The PSESWG Model [25] provides a list of the capabilities a PSE may provide. Since a SEE is a specific type of PSE, the PSESWG Model defines the capabilities a SEE might provide. Section 3.2 provides an overview of the PSESWG Model and Appendix A provides a description of the PSESWG Model services.

Mapping all the tools compatible to a SEE framework to the PSESWG Reference Model shows all possible ways the user can configure the SEE and the range of functionality available for the SEE. There may be several tools which correspond to a specific service or there may be no tools corresponding to a particular service. Many services described in PSESWG are not necessary for every PSE and the assessor cannot expect every SEE to provide every service; however, the mapping will show what tool capabilities are available for a particular SEE framework. Figure 4 illustrates mapping the SEE tools to the PSESWG Reference Model.

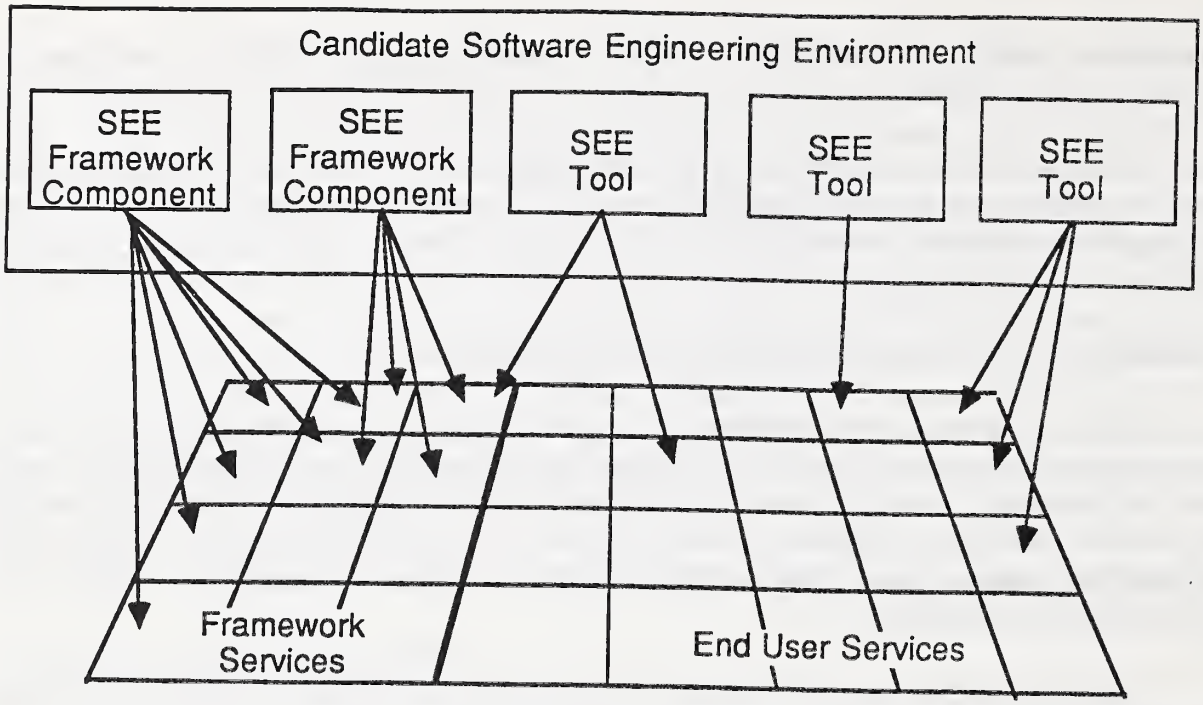

Figure 4: Mapping Candidate SEEs to the PSESWG Model

The reference model mapping will not indicate anything about the relationship of tools of the same type to each other. While two tools may provide the same basic service, they may provide it very differently or support different integration standards. For example, an ANSI C Compiler and a LISP Interpreter will both provide the capabilities of PSESWG's Software Generation Service, but they cannot substitute for each other. The assessor can compare tools of the same type using one of the models of tool assessment and evaluation available. Appendix B provides two examples of tool evaluation methods, and Appendix D provides sources of criteria for assessing individual tools. The SEE assessment process assesses only the general capabilities of the SEE, not the details of the individual tools.

The results of mapping the PSESWG Model to the SEEs should be a list of the tools compatible to each SEE grouped according to the functions those tools provide as defined in the PSESWG Reference Model.

4.7 Results of Assessment Process

The result of the overall assessment process for each identified SEE framework is an assessment of the SEE in three areas provided by information generated during the process. These areas and the information follow in Figure 5:

1) **Structure:** Mapping of the SEE framework to the NIST/ECMA Framework Model including
    a) identification of possible SEE frameworks built from available components
    b) identification of the capabilities provided
    c) identification of the capabilities *not* provided
    d) analysis of how separable framework components are from the tools
2) **Functionality:** Mapping of the tools available for the SEE to the PSESWG Reference Model
    a) identification of the options available for populating the SEE
    b) identification of the range of functionality provided for the SEE
    c) identification of functionality not available for the SEE
3) **Integration:** Identification of interfaces and interface standards for the framework
    a) identification of standards compliance
    b) analysis of the extent of integration supported by framework

Figure 5: Results of Assessment Process

To summarize, the purpose of assessing a SEE is to describe accurately with reference to an accepted model what the capacity of a SEE is to support and integrate a range of CASE tools. This assessment process describes the SEE's structure by mapping the SEE framework to the NIST/ECMA Framework Model to isolate and define the framework capabilities of the SEE. The assessment process describes the integration possible in the SEE by identifying the type and extent of framework interfaces for the SEE. Finally, the assessment process describes the functionality which the customer can add to the SEE by mapping the tools available for the SEE to the PSESWG Reference Model.

Section 5: Software Engineering Environment Evaluation Methodology

As was stated earlier, the purpose of evaluating a SEE is to establish how well a SEE meets the customer organization's needs. Those needs include both immediate functional needs in support of existing practice and long term organizational goals for evolving practices. A SEE supports a strategy for satisfying both. For an organization to find a SEE evaluation useful, the evaluation must relate the organization's existing and desired practices to the SEE's capabilities. Therefore, a SEE evaluation should include establishing what the using organizations needs are, what capabilities individual SEEs have and how well these SEEs can be expanded or upgraded in the future to meet expected future needs of the organization [26].

5.1 Overview of the Software Engineering Environment Evaluation Process

The evaluation process builds on the assessment process by examining the SEE capabilities in greater detail and relating them to the customer organization's needs. The assessment process considers three characteristics of SEEs: structure, integration and functionality. The evaluation process broadens this examination of SEEs by establishing what characteristics are important to the SEE *customer* and applies this information to testing the SEEs for their applicability to the customers needs. The evaluation process uses the assessment process as a starting point to initially assess the capabilities of available SEEs. The evaluation process tailors that assessment to a particular organization. An overview of the steps of the evaluation process and what each step produces follows in Figure 6:

1) Establish customer requirements
   Define customer's software engineering methodology
   Define how customer wants SEE to support methodology
   Define how customer expects SEE to change with evolving methodology
   Result: Statement of how customer expects the SEE to support current and evolving software development methodology in organization
2) Select criteria for evaluating candidate SEE components from customer's requirements
   Define test plan for criteria
   Result: List of criteria for evaluating SEEs matching customer's requirements and plan for testing each criteria
3) Use Assessments to Select Candidate SEEs
   Result: Assessments of SEEs defining SEE structure, integration and range of possible functionality
4) Execute test plan on candidate SEEs
   Result: Report of how candidate SEEs meet each criteria
5) Analyze test plan results
   Result: Analysis of how well candidate SEEs are likely to meet customer organization's requirements and ranking of candidate SEEs

Figure 6: SEE Evaluation Process

This model draws on IEEE 1209, Recommended Practice for Tool Evaluation [23], the Draft ISO Recommended Practice for Tool Evaluation and Selection [11] and the STSC model of tool evaluation [14]. Appendix B describes these models of tool evaluation.

5.2 Establish Customer Requirements

5.2.1. Prerequisites for effective SEE use

In order for an organization to effectively use a full SEE or a SEE supporting a large part of the software development process, the organization must have a commitment to using a software development methodology and have a clear role for the SEE in support of that process. This requires the organization to have a clear definition of what its software development process is. Many organizations do not have this knowledge [26]. A SEE supporting a software development process which is new to an organization or not strongly supported by management will not provide the desired productivity gains. Instead, the SEE will probably be abandoned [16]. New tools and methodologies will not by themselves bring order to a chaotic software development process and will not help in on-going projects which are time critical [26]. The SEI CMM Model does not take into account an organization's tool usage for establishing its software development maturity level until level 4 [20]. This late consideration reflects the need for organizations to establish good practice before selecting tool support. Learning a new methodology and learning to use CASE tools are both time and capital intensive efforts requiring long term management commitment [16, 20, 26].

Because a new SEE represents a substantial organizational commitment in capital and training time, a SEE should be usable over a long period of time. Tools, hardware, software development methodology and organizational practices will change during that time. Therefore, the organization should have long-term goals for the evolution of their software development activities and a sense of where the larger software engineering community and SEE market is heading. If the organization does not select a SEE with a view to its future needs in software development support, the SEE could become obsolete as the organization changes or new technology emerges. The SEE chosen should be one that can be upgraded with new tools to meet the organization's evolving needs.

5.2.2 Define how customer expects the SEE to support processes

The customer must establish what SEE functions he can effectively use to support the software engineering process and software development methodologies of his organization. As a prerequisite to this evaluation process, the customer organization must establish what the software development process and methodologies of the organization are and what corporate goals exist for evolving the process. Only with that information can the customer decide on the role the SEE will have in support of the process and methodology and decide what SEE capabilities the organization needs. How the organization views the SEE's role in its software development process affects current requirements for a SEE and expectations about how the SEE will evolve in combination with the process [26].

The NIST/ECMA and PSESWG Models provide a format for defining these functions. The customer should specify desired functions in terms of the reference model services and attach any specific requirements onto the appropriate service dimensions. The customer can also rate the services or requirements on the service in terms of criticality to the organization's goals. The reference models are useful for defining service requirements because they provide standard definitions of capabilities in a standard format without prejudging the evaluation process with the definition of a specific architecture.

While the reference models can help the customer define functional requirements, the customer must also define integration requirements for the SEE. The customer's integration requirements reflect his view of how the SEE should evolve with the software development process. Specifically, if the customer wants the SEE to be an expandable integrated "open system" allowing multiple vendors to supply products in the future, the SEE framework must meet "open standards". The customer will want the SEE to conform to those integration standards allowing him the greatest flexibility and room for evolving and expanding his SEE in the future; otherwise, his commitment in time and money may be wasted if critical parts of the SEE become obsolete.

The customer should define his integration requirements in terms of standards or product compatibility. Using the five integration types [28] and five integration levels discussed in Section 3.3, the customer can specify the minimum level of integration, the customer will accept in any one area. By specifying actual standards or product compatibility, the customer is providing his view of where industry is going.

Reference models are a starting point for describing customer requirements and environment characteristics. The customer will have requirements on SEE function not captured in the reference models, and the SEEs will have characteristics not in the models. For example, the customer may have requirements about cost or future product support. Other possible customer requirements not otherwise captured may include hardware or software compatibility requirements, performance requirements or cost limitations.

5.2.3 Results of establishing customer requirements

Establishing the customer requirements should result in a written document defining exactly what the customer expects from an SEE and how the customer expects to incorporate the SEE into the organization including the following:

1) Organizational functions the SEE should support
2) Relationship of the SEE to the organization's existing methodology
3) SEE capabilities in terms of reference model services and dimensions
4) SEE integration in terms of standards or types and levels
5) Labor and capital available to invest in purchase and training
6) Hardware and software compatibility requirements
7) Any additional requirements

If the evaluators know where and how an organization expects to use a SEE and what the organization expects the SEE to do, the evaluators can have a better sense of what SEE functions the organization needs. Knowing an organization's current methods and expectations of a SEE provides the evaluator with the information he needs to relate SEE capabilities to the customer requirements. The evaluator can also judge if the market has any product matching the customers expectations. Defining the organization's requirements and the SEE capabilities in terms of the reference model services and dimensions gives a set of standard definitions that the evaluator can refer to when looking at the capabilities of candidate SEEs.

5.3 Select Criteria for Evaluation

5.3.1 Identify criteria

In the next step of the SEE evaluation process, the evaluator takes the customer's requirements and creates a list of criteria for evaluating the SEE reflecting those requirements. By grouping both the customer requirements and the evaluation defined criteria using the reference model services and dimensions, the evaluator can easily map the customer requirements to the evaluation criteria providing a check that the customer and evaluator agree on the important elements to look for in a SEE. Grouping criteria by service or service group is easier than grouping criteria by type of tool since tools provide a variety of combinations of capabilities and service choices should not imply an architecture. Because a candidate SEE is a complex package of software tools providing a variety of capabilities, it is also easier to relate the tools to criteria grouped by services. Taking the customer's requirements as a starting point, the evaluator must make sure that he has selected criteria which will check that the candidate SEEs meet all the customer's stated requirements and allow for comparison of the candidate SEEs.

There have been a number of published sets of detailed criteria for evaluating tools which the SEE evaluator may find helpful. STSC has published sets of criteria for evaluating several different types of tools including testing tools and design tools. IEEE 1209 and the draft ISO guide include lists of criteria for evaluating different types of tools. Appendix D discusses these and other criteria sources. However, there are no equivalent recognized lists of criteria for integration. Currently, the source for integration criteria is the customer's own requirements for compatibility between the tools and the SEE. For evaluating individual tools the lack of integration criteria is not as important an issue as it will be in the future when more SEEs are available and customers are looking for tools compatible to their SEE.

Because the SEE industry is new and volatile, the customer's stated requirements may not provide enough criteria to illustrate important differences between competing SEEs. The customer's stated requirements may not include consideration of future industry direction or likely adoption of competing standards. Because SEEs are new products, standards, especially integration standards, are still evolving. The customer also may not have considered important issues such as standards compliance and likely upgrades.

The evaluator should present a written description of the criteria for assessing SEEs to

the customer. The description should contain the following:

1) Criteria for each customer requirement on a SEE capability
2) Integration criteria mapped to the integration requirements
3) General criteria mapped to the general requirements

The evaluator should clearly define the criteria and demonstrate the relationship of the criteria to the requirements.

5.3.2 Create test plan for criteria

Once the evaluator has selected the criteria, he must decide how to apply the criteria to the candidate environments and how he should record the results of applying the criteria. How the evaluator applies the criteria depends on what the criteria are. The STSC tool evaluation studies [e.g., 14, 15] provide one suggested method for recording test results. The evaluator should test some criteria, such as performance or integration criteria, by actually trying the SEE or tools. The evaluator may be able to apply other criteria by reading the documentation and other vendor literature. The results of applying the criteria might be a specific objective value such as the languages supported by the tool or it might be a subjective value such as the evaluator's rating of the tool support available. Integration criteria should be tested by actually testing if the tools can function cooperatively to the desired extent.

The evaluator should present the results of this step of the evaluation process as a written test plan for evaluating the candidate SEEs with respect to the criteria. The test plan should consist of the proposed mechanism for testing each criteria and the scale or values associated with each criteria. For objective criteria, the proposed mechanism should be a repeatable procedure always resulting in the same value (e.g., a benchmark test for performance of a program on a given hardware and software platform). For subjective criteria, the proposed testing mechanism should explicitly define how the evaluators will assign values to the criteria. The test plan should distinguish among criteria which have to be tested on the full SEE, criteria which can be tested on individual components and criteria which require only reading the documentation.

5.3.3 Results of criteria selection step

The evaluator should present the results of criteria selection step in two parts. The first is a list of criteria for evaluating candidate tools and environments sorted by service when possible. The second is a mapping from the customer's requirements to the criteria, showing that the criteria cover the customer's requirements.

5.4 Use Assessments to Identify Candidate SEEs

The evaluator must identify the frameworks and software development tools comprising candidate SEE environments and describe them in terms of the reference models. A description

of the reference model services the SEE provides allows the evaluator to make some preliminary selections of candidate SEEs to evaluate.

The assessment process provides lists of candidate frameworks and the tools which are compatible to each framework. The assessment process describes each framework component or tool in terms of the reference model services it provides and the integration mechanisms it uses. The evaluator should have organized the customer's requirements and the evaluation criteria in terms of the reference model services, the type of integration required and any general criteria. Each candidate SEE consists of the minimal set of framework components and tools providing the required services. To identify the candidate SEE environments, the evaluator must identify which tools and frameworks are compatible and which tools perform the same functions. If a number of tools conform to the same standards, the customer may be able to combine the tools in a number of different candidate SEEs. For example, there may be several different design tools which all work with the same SEE repository. All the different configurations are separate candidate SEEs. Figure 7 illustrates the mapping of the candidate SEEs to minimally provide the set of required services. An 'R' indicates a required service in the diagram.

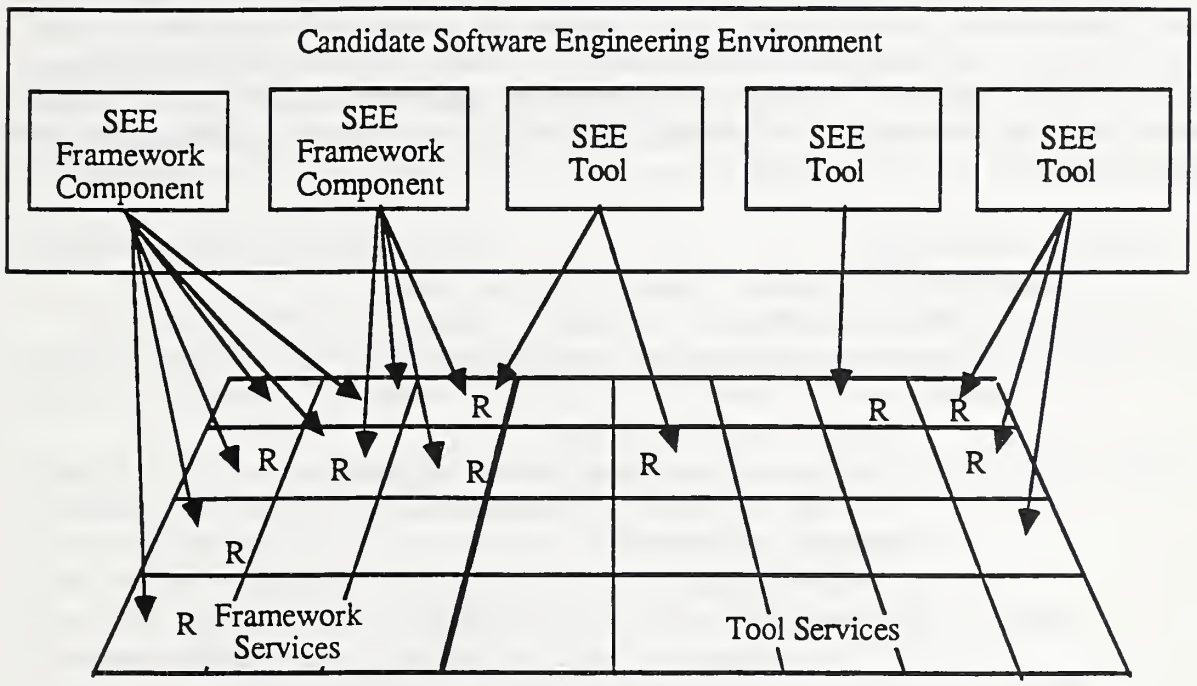

Figure 7: Identifying a Candidate SEE

The framework components and tools making up a candidate SEE may be from one or several vendors. To ensure that all the pieces can function as a SEE, it is important for the evaluator to at least see the combination demonstrated and preferably test it himself. Use of the same integration standards is *not* a guarantee of compatibility.

The result of this step is the list of candidate SEEs and their components for evaluation. The evaluation process identifies the candidate SEEs by identifying unique combinations of compatible SEE frameworks and tools apparently providing the required services at a minimum level. Because the same SEE component or tool may be part of several candidate SEEs, the evaluator should do as much independent tool testing as possible and only test the tool as part of the SEE when absolutely necessary. By separating the testing of each tool, many of the evaluation results for each tool are available for reuse.

5.5 Execute Plan for Testing Criteria on Candidate SEEs

After identifying the candidate SEEs, the evaluator can execute the test plan developed in step 2 of the evaluation process on each candidate SEE. Only one evaluator is necessary for evaluating objective criteria since an easily checked, repeatable procedure exists. However, several evaluators should apply specific standards to test the subjective criteria.

Like the customer requirements and the evaluation criteria, the evaluators should structure the test plan in terms of the reference model services. For any specific SEE component, the criteria and test plan applicable to that component depend on the services that the component provides. Since a candidate SEE is a SEE providing all of the services mentioned in the customer's requirements, each criterion will be tested on at least one component of each candidate SEE. Because SEE components may be part of more than one SEE, the evaluators should maintain the results of testing each component separately, assembling the various component tests into overall SEE evaluations as appropriate. The algorithm for testing SEEs follows in Figure 8:

```
for each candidate SEE
        execute tests for criteria on candidate SEE as a whole
        for each SEE component
                if it has been tested before, retrieve past test
                else
                        find the services it maps to
                        find the criteria applying to those services
                        find the test plan(s) for those criteria
                        execute the test plan
                        save the results
        collect all SEE component testing results together
```

Figure 8: Algorithm for Applying Criteria to Candidate SEEs

Because both the SEE components and the criteria relate to the reference model services, the evaluator can use the reference model services to select the tests applicable to specific SEE components. The test plan should also separate tests for individual tools and for the full SEE. For example, performance for the full SEE may differ substantially from the performance of an individual component.

The results of this step are a series of matrices listing the value for each criterion for each candidate SEE or SEE component.

5.6 Analyze Results of Testing Criteria

The record of the results of executing the test plan provides only the raw data. The evaluator must interpret this data for the customer. The customer needs to know how well each candidate SEE satisfies his requirements, how well it can support his software development methodology, what inherent limitations in each SEE exist, how those limitations will affect the SEEs long term usefulness and how the SEE compares to other SEEs.

While the raw data resulting from testing the criteria provides the details of how the SEE satisfies the requirements, the customer needs to have a broader view before buying a product. He needs a tailored summary of the results highlighting what is important to the customer. For example, the customer needs to know about available tool support for the SEE which would support his software development methodology. The customer also needs to know about any inherent limitations of the framework uncovered in the assessment process limiting the integration possible in the SEE. Finally, the customer needs to know how each candidate SEE compares to the others.

The results of this step should be an easily understood summary of how each candidate SEE relates to the actual needs of the customer organization, how the SEEs compare, what their inherent limitations are and which SEE provides the best fit to the customer's software development process. The summary should reinforce the data generated in the previous step.

5.7 Results of the Evaluation Process

The results of the evaluation process are a series of documents transforming the customer's needs into testable criteria and transforming the results of testing SEEs into a usable form for the customer. The results of the evaluation process follow in Figure 9:

1) Description of customer's software development methodology and expected role of SEE
2) Customer requirements for SEE
3) Evaluation criteria for testing whether the SEEs meet the requirements
4) Test plan for evaluating SEEs with respect to the criteria
5) Results of each test on each SEE
6) Summary relating results generated in 5 back to requirements from 1

Figure 9: Results of Evaluation Process

To summarize the purpose of evaluating SEEs is to establish how well a SEE meets the customer organization's needs. By providing the information on how the evaluator transforms information about the customers needs into testable criteria and what the results of testing the criteria are, the customer can see how the evaluator justified the conclusions. In addition, by

organizing the requirements, criteria for testing and description of the SEEs in terms of the reference model services and integration criteria, tracing information related to a defined capability is easier.

5.8 Performing Evaluations

All watchers of the CASE tool marketplace stress the rapidly evolving nature of that market. There are new products or updated products coming out frequently. In the SEE area, there are few SEE frameworks actually available, but many in the development stage. There are also vendors marketing groups of compatible tools or toolsets which do not use a framework. How these products can be extended in the future is unclear. It is also unclear how compatible many of the planned SEE frameworks and tools for frameworks actually will be.

Because of this volatility, SEE evaluations and assessments reflecting the state of the marketplace must be completed quickly or risk being obsolete before use. Alternatively, the customer may wish to perform periodic surveys of the available products and not invest in extensive evaluations or assessments until the technology is more mature.

Section 6. Conclusions

Defining assessment and evaluation processes for SEEs, SEE frameworks and CASE tools at this time is a difficult task because the market for these products and the products themselves are still evolving. While the problem of integrating CASE tools into SEEs is well recognized, the solution integration standards and methods have not been agreed on. The customer in this market needs to recognize its volatility and assess products for future viability in addition to present needs.

This guide has presented a general approach to SEE assessment and evaluation focusing on accurately defining the integration and functional capabilities of candidate SEEs and the requirements of the customer. The assessment process concentrates on accurately representing the functional and integration capabilities of SEEs in terms of the NIST/ECMA Framework Model, the NGCR PSESWG Model and accepted integration definitions. The evaluation process builds on the assessment process by representing the customer's requirements in terms of the same definitions and relating the customers requirements to the capabilities of the candidate SEEs. Thus, the evaluation process builds on the assessment process.

This is a general approach to SEE assessment and evaluation only. The approaches should be tested by actually applying them to a real organization's needs and a group of candidate software engineering environments.

Section 7. References

[1] Berk, Kevin, Dean Barrow, and Todd Steadman. Project Management Tools Report. Software Technology Support Center, March, 1992.

[2] Bowler, Odean, John Grotzky, Mark Nielson, Susan Nilson and Jim Van Buren. Requirements Analysis and Design Tools Report. Software Technology Support Center, April 1992.

[3] Brown, Alan W. An Approach Toward the Selection of Data Interface Standards. Draft Technical Report from Next Generation Computer Resources Project, 1992.

[4] Brown, Alan W., David J. Carney, Peter H. Feiler, Patricia A. Oberndorf and Marvin V. Zelkowitz. Issues in the Definition of a Project Support Environment Reference Model in Annual Report of the SEI, 1993.

[5] Brown, Alan W., Anthony N. Earl and John A. McDermid. *Software Engineering Environments: Automated Support for Software Engineering*. New York, McGraw-Hill Book Co., 1992.

[6] Brown, Alan W. and John A. McDermid. Learning from IPSE's Mistake. In IEEE Software, Vol. 9, No.2, March 1992.

[7] Chen, Minder and Ronald J. Norman. "A Framework for Integrated CASE" from *IEEE Software* Vol. 9, No. 2, March, 1992.

[8] Chikofsky, Elliot J., David A. Martin and Hugh Chang. "Assessing the State of Tools Assessment" from IEEE Software, Vol. 9, No.3, May, 1992.

[9] Crosby, Davie, Gary Petersen and Reed Sorenson. Documentation Tools Report. Software Technology Support Center, March, 1992.

[10] Draft Guide to the POSIX Open Systems Environments. IEEE Std. P1003.0. 1992.

[11] Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1992.

[12] Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems Inc., 1993.

[13] Firth, Robert, Vicky Mosley, Richard Pethia, Lauren Roberts, and William Wood. A Guide to the Classification and Assessment of Software Engineering Tools. Carnegie Mellon University Software Engineering Institute, Technical Report #CMU/SEI-87-TR-10

[14] Hanrahan, Bob, Ron Peterson, Judi Peterson and Dennis Barney. Software Engineering Environment Report. Software Technology Support Center, March, 1992.

[15] Horgan, Joseph R. and Aditya P. Mathur. "Assessing Testing Tools in Research and Education" from IEEE Software, Vol. 9, No.3, May, 1992.

[16] Kemerer, Chris F. "How the Learning Curve Affects CASE Tool Adoption" from IEEE Software, Vol. 9, No.3, May, 1992.

[17] McGary, F. and R. Pajushi. "Towards Understanding Software - 15 Years at the SEI" from the 15th NASA/6th SFC Software Engineering Workshop, Nov. 1990. SEI TR 90-006.

[18] Mosley, Vicky. "How to Assess Tools Efficiently and Quantitatively" from IEEE Software, Vol. 9, No.3, May, 1992.

[19] NIST Framework Model Mapping Guidelines, Version 1.2. (Draft) National Institute of Standards and Technology, 1991.

[20] Paulk, Marc C., Bill Curtis, et al. Capability Maturity Model for Software. Carnegie Mellon University Software Engineering Institute, CMU/SEI-91-TR-24, 1991.

[21] Poston, Robert M. and Michael P. Sexton. "Evaluating and Selecting Testing Tools" from IEEE Software, Vol. 9, No.3, May, 1992.

[22] Price, Gordon, Bryce Ragland, Daren Murdock and Eric Hidden. Source Code Static Analysis Tools Report. Software Technology Support Center, April 1992.

[23] Recommended Practice for the Evaluation and Selection of CASE Tools. IEEE Standard 1209, 1993.

[24] Reference Model for Frameworks of Software Engineering Environments, Edition 3. (Draft) National Institute of Standards and Technology, 1993.

[25] Reference Model for Project Support Environments, Version 1. Technical Report, NAWC-ADWAR,-93023-70, 1992.

[26] Smith, Dennis, Cliff Huff, Ed Morris, and Paul Zarrella. Software Engineering Environment Evaluation Issues. Carnegie Mellon University Software Engineering Institute, Technical Report, 1993.

[27] Thomas, Ian and Brian A. Nejmeh. "Definitions of Tool Integration for Environments" from IEEE Software, Vol. 9, No.2, March, 1992.

[28] Wasserman, Anthony I. "Tool Integration in Software Engineering Environments" from

Lecture Notes in Computer Science: Software Engineering Environments ed. by Fred Long. New York: Springer-Verlag, 1989.

[29] Zelkowitz, Marvin V. "Use of an Environment Classification Model" ACM/IEEE 15th International Conference on Software Engineering. Baltimore, MD, May 1993 (to appear).

Appendix A: Definitions of NGCR PSESWG Reference Model Services and NIST/ECMA Framework Model Services

This appendix contains a list of the services and service groups definitions found in the NGCR PSESWG Reference Model for Project Support Environments [25] and the NIST/ECMA Reference Model of Frameworks of Software Engineering Environments [24]. Because the NGCR PSESWG Model incorporates the NIST/ECMA Framework Model, the organization of the service definitions reflects that of the NGCR PSESWG Model. The definitions given paraphrase those in [6]. The number in parenthesis after the definition refers to the section of the NGCR PSESWG Model defining the service or service group.

A.1 Technical Engineering Services

System Engineering Services (4.1) - These services support projects involving substantial development or maintenance of hardware and software components. These services integrate the results of and provide consistency between the more specialized engineering services (i.e., software engineering services).

System Requirements Engineering Service (4.1.1) - This service provides the capabilities to capture, model, analyze, represent and refine the system requirements for a system containing some combination of software, hardware, facilities, people and data. This service creates and manipulates representations of system requirements.

System Design and Allocation Service (4.1.2) - This service provides the capabilities to create an architectural design of a system's components including the interrelationships of system components, the partitioning of system functionality, and constraints between hardware and software.

System Simulation and Modeling Service (4.1.3) - The ability to model (or prototype) a system concept in its entirety before implementation takes place.

System Static Analysis Service (4.1.4) - This service provides for the static analysis of system designs and components in order to determine the attributes of the system.

System Testing Service (4.1.5) - This service supports the testing of systems to insure that all specifications have been met and that systems are operationally effective and suitable for intended use.

System Integration Service (4.1.6) - This service supports the integration of the different pieces of a project into a single product.

System Re-Engineering Service (4.1.7) - This services supports the modification of an existing

design as a response to a changed set of requirements.

Host-Target Connection Service (4.1.8) - This service ensures the ability of a host PSE to communicate with a target system for the purpose of software downloading, system test or debug, and system monitoring.

Target Monitoring Service (4.1.9) - This service provides the ability of the host PSE to receive and interpret specified execution and performance information from an operational target system.

Traceability Service (4.1.10) - This service supports the recording of relationships between artifacts of the development process.

Software Engineering Services (4.2) - These services support the specification, implementation, debugging and maintenance of software.

Software Requirements Engineering Service (4.2.1) - This service provides the capabilities to capture, represent, analyze and refine those system requirements allocated to software components. This service creates and manipulates representations of requirements.

Software Design Service (4.2.2) - This service provides the capability to create a design of the software components of a system or subsystem.

Software Simulation and Modeling Service (4.2.3) - The ability to model (or prototype) a software design before full implementation takes place.

Software Verification Service (4.2.4) - This service supports the formal verification of software against its formal specifications for the purpose of locating errors.

Software Generation Service (4.2.5) - This service provides semi-automatic production of software using existing components or templates.

Software Compilation Service (4.2.6) - This service supports the translation and linking of software components written in various programming languages.

Software Static Analysis Service (4.2.7) - This service provides for the static analysis or source code analysis of software components.

Software Debugging Service (4.2.8) - This service supports the location and repair of software errors in individual software components by controlled or monitored execution of code.

Software Testing Service (4.2.9) - This service supports the testing of software systems.

Software Build Service (4.2.10) - This service supports the integration of separately developed

software components into a single system.

Software Reverse Engineering Service (4.2.11) - This service provides the capabilities to capture design information from source or object code and produce structure charts, call graphs and other design documentation.

Software Re-Engineering Service (4.2.12) - This service is used to modify an existing design to conform to a new set of requirements.

Software Traceability Service (4.2.13) - This service supports the recording of relationships between artifacts of the software development process.

Life-Cycle Process Engineering Services (4.3) - These services support projects in achieving discipline and control over their life-cycle development processes and individual process steps.

Process Definition Service (4.3.1) - This service provides the capabilities for projects to create, maintain, tailor, adapt and validate definitions of processes in formal, semiformal and informal forms.

Process Library Service (4.3.2) - This service supports reuse capabilities for processes.

Process Exchange Service (4.3.3) - This service supports the interchange of process definitions between projects and PSEs.

Process Usage Service (4.3.4) - This service supports the execution of a project's defined and installed process.

A.2 Technical Management Services

Configuration Management Service (5.1) - This service supports the identification, documentation and control of the functional and physical characteristics of configuration items to ensure traceability and reproducabiltiy of a project's end products.

Change Management Service (5.2) - This service supports the creation, evaluation, and tracking of change requests generated in response to errors, omissions or required enhancements to a product.

Reuse Management Service (5.3) - This service supports the storage, inspection, and reuse of assets related to many stages of engineering processes.

Metrics Service (5.4) - This service supports the collection and organization of qualitative data

about the PSE's processes and products.

A.3 Project Management Services

Scheduling Service (6.1) - This service supports the handling of data according to a set of chronological constraints relevant to a project.

Estimating Service (6.2) - This service supports quantification, analysis and prediction of project cost and resource needs.

Risk Analysis Service (6.3) - This service supports those planning activities that consider elements related to the success or failure of a project.

Tracking Service - This service supports correlation of estimated cost and schedule data with actual performance of a project.

A.4 Support Services

Common Support Services (7.1) - These services support standard representations for communication among PSE customers.

Text Processing Service (7.1.1) - This service provides the ability to create and manipulate textual information within the PSE.

Numeric Processing Service (7.1.2) - This service provides the ability to create and manipulate numeric information.

Figure Processing Service (7.1.3) - This service provides the ability to create and manipulate graphic, image or documentation figures.

Audio and Video Processing Service (7.1.4) - This service provides the ability to capture, create and manipulate data from audio-or video-based sources.

Calendar and Reminder Service (7.1.5) - This service provides the means for a user to keep an electronic schedule of meetings, deadlines, and similar important dates and times.

Annotation Service (7.1.6) - This service provides for associating comments with existing objects.

Publishing Service (7.2) - This service provides the capabilities necessary to create and print documents.

Presentation Preparation Service (7.3) - This service provides the capabilities necessary to produce materials for presentation.

User Communication Services (7.4) - These services support interaction between PSE users.

Mail Service (7.4.1) - This service provides for passing notes between computer system users.

Bulletin Board Service (7.4.2) - This service supports the capabilities normally associated with an electronic bulletin board.

Conferencing Service (7.4.3) - This service supports interactive synchronous communication among users.

PSE Administration Services (7.5) - These services support the administration of the PSE.

Framework Administration and Configuration Services (7.5.1) - A SEE framework has to be carefully administered because its precise configuration may be constantly changing to meet the changing needs of the software development enterprise. These services provide for general framework administration. (These services taken from the NIST/ECMA Framework Model [24])

Tool Registration Service (7.5.1.a) - This service provides a means for incorporating new tools into an environment based on the framework in such a way that different framework components coordinate effectively with the new tool.

Resource Registration and Mapping Service (7.5.1.b) - This service provides the means for managing, modelling and controlling the physical resources of the environment.

Metrication Service (7.5.1.c) - This service provides the ability to collect technical measurement information of importance to the administration of the framework.

User Administration Service (7.5.1.d) - This service provides the ability to add users to an environment, to characterize their modes of operation and roles (including security privileges), and to make available to them the resources which they require.

Self-Configuration Management Service (7.5.1.e) - This service supports the existence of many simultaneous coresident configurations of a framework implementation.

Tool Installation and Customization Service (7.5.2) - This service supports the installation,

testing and registration of tools into a PSE.

PSE User and Role Management Service (7.5.3) - This service provides the ability to make users of a PSE known to the system.

PSE Resource Management Service (7.5.4) - This service provides the ability to monitor, add, change or delete resources available to a PSE.

PSE Status Monitoring Service (7.5.5) - This service provides the ability to monitor and control the actions taking place.

PSE Diagnostic Service (7.5.6) - This service provides the capabilities for the PSE to perform self-testing and diagnosis of error conditions.

PSE Interchange Service (7.5.7) - This service supports communication and data sharing between PSEs.

PSE User Access Service (7.5.8) - This service provides the capability of monitoring who is accessing resources and provides control over access to the PSE.

A.5 Framework Services

Operating System Services (8.1) - These services support what are generally considered operating system capabilities. (These services taken from IEEE P1003.0 [10])

Process Creation and Deletion Service (8.1.a) - This service supports the creation and deletion of system processes.

Process Attribute Service (8.1.b) - This service provides the ability to determine attributes of system processes.

Node Internal Communication and Synchronization Services (8.1.c) - This service handles supports the accessing and manipulation of data at the operating system level.

Generalized Input and Output Services (8.1.d) - This service provides access to device drivers.

File-Oriented Services (8.1.e) - This service provides for the organization and manipulation of files.

Event, Error and Exception Management Services (8.1.f) - This service provides the ability to handle asynchronous events in a system

Time Services (8.1.g) - This service provides for the manipulation of system timers.

Memory Management Services (8.1.h) - This service provides for the allocation of virtual and fixed memory in a system.

Logical Naming Services (8.1.i) - This service renames system resources by logical names rather than physical names.

Resource Management Services (8.1.j) - These services provide for general system management.

Object Management Services (8.2) - The general purpose of the object management grouping is the definition, storage, maintenance, management, and access of object entities and the relationships among them. (These services taken from the NIST/ECMA Framework Model [24])

Metadata Service (8.2.a) - This service provides definition, control, and maintenance of metadata (e.g., schemas), typically according to a supported data model.

Data Storage Service (8.2.b) - This service provides definition, control, and maintenance of objects, typically according to previously defined schemas and type definitions.

Relationship Service (8.2.c) - This service provides the capability for defining and maintaining relationships between objects in the object management system. It may be an intrinsic part of the data model or it may be a separate service.

Name Service (8.2.d) - This service supports naming objects and associated data and maintains relationships between surrogates and names.

Distribution and Location Service (8.2.e) - This service provides capabilities that support management and access of distributed objects.

Data Transaction Service (8.2.f) - This service provides capabilities to define and enact transactions.

Concurrency Service (8.2.g) - This service provides capabilities that ensure reliable concurrent access (by users or processes) to the object management system.

Operating System (OS) Process Support Service (8.2.h) - This service provides the ability to define OS processes (i.e., active objects) and access them using the same mechanisms used for objects, i.e., integration of process and object management.

Archive Service (8.2.i) - This service allows on-line information to be transferred to off-line media and vice-versa.

Backup Service (8.2.j) - This service restores the development environment to a consistent state after any media failure.

Derivation Service (8.2.k) - This service supports definition and enactment of derivation rules among objects, relationships or values (e.g., computed attributes, derived objects).

Replication and Synchronization Service (8.2.l) - This service provides for the explicit replication of objects in a distributed environment and the management of the consistency of redundant copies.

Access Control and Security Service (8.2.m) - This service provides for the definition and enforcement of rules by which access to SEE objects (e.g., data, tools) may be granted to or withheld from user and tools.

Function Attachment Service (8.2.n) - This service provides for the attachment or relation of functions or operations to object types, as well as the attachment and relation of operations to individual instances of objects.

Common and Canonical Schema Service (8.2.o) - This service provides mechanisms for integrating tools into a SEE by providing a means to create common (logical) definitions of the objects (and operations) these tools may share from the underlying objects in the OMS.

Version Service (8.2.p) - This service provides capabilities for managing data from earlier states of objects in the OMS. Change throughout development has to be managed in a SEE, and the inclusion of versioning is one of the means of achieving this.

Composite Object Service (8.2.q) - This service creates, manages, accesses, and deletes composite objects, i.e., objects composed of other objects. It may be an intrinsic part of the data model or a separate service.

Query Service (8.2.r) - This service is an extension to the data storage service's "read" operation. It provides capabilities to retrieve sets of objects according to defined properties and values.

State Monitoring and Triggering Service (8.2.s) - This service enables the specification and enaction of database states, state transformations, and actions to be taken should these states occur or persist.

Sub-Environment Service (8.2.t) - This service enables the definition, access, and manipulation of a subset of the object management model (e.g., types, relationship types, operations if any) or related instances (e.g., actual objects).

Data Interchange Service (8.2.u) - This service offers two-way translation between data repositories in different SEEs.

Policy Enforcement Service (8.3) - The reference model uses the term ''policy enforcement'' to cover the similar functionality of security enforcement, integrity monitoring, and various object management functions such as configuration management. The SEE reference model regards security as a service that crosses many of the boundaries of the reference model divisions. (These services taken from the NIST/ECMA Framework Model [24])

Mandatory Confidentiality Service (8.3.a) - This service provides a means of implementing those policies established by an administrator concerning access to the information contained in an object.

Discretionary Confidentiality Service (8.3.b) - This service provides a means of implementing those policies established by a user concerning access to the information contained in an object and becomes largely a matter of personal privacy.

Mandatory Integrity Service (8.3.c) - This service provides assurance that a system object maintains (or at least tracks) the ''purity'' or ''goodness'' of an object by recording exactly what has been done to the object and how it was done.

Discretionary Integrity Service (8.3.d) - This service provides for controls implemented by all write, modify, and append permission functions defined for discretionary access controls.

Mandatory Conformity Service (8.3.e) - This service provides a means of automating operational models.

Discretionary Conformity Service (8.3.f) - This service provides a means for individual users to structure their own work environment. Under the right conditions, it could turn out to be the equivalent of ''canned procedures'' or ''command scripts.''

Process Management Services (8.4) - The general purposes of the Process Management Services in a SEE are the unambiguous definition and the computer-assisted performance of software development activities across total software lifecycles. In addition to technical development activities, these potentially include management, documentation, evaluation, assessment, policy-enforcement, business control, maintenance, and other activities. (These services taken from the NIST/ECMA Framework Model [24])

Process Definition Service (8.4.a) - This service provides a means of defining new process assets which may be complete processes, subprocesses or process architectures for a process library.

Process Enactment Service (8.4.b) - This service provides a means of enacting processes using human or machine process agents.

Process Visibility and Scoping Service (8.4.c) - This service provides a means for several enacting process elements to cooperate to achieve the goals of a larger process. Logically, the extent of such cooperation is part of the definition of processes and may be provided by integrated visibility and scoping features with the process definition service.

Process State Service (8.4.d) - This service provides a means of maintaining the changing "process state". Certain changes in the enactment state of a process may be defined as ''events'' and may act as conditions or constraints affecting other processes.

Process Control Service (8.4.e) - This service provides a means of recording, measuring, controlling, managing or constraining the enacted processes.

Process Resource Management Service (8.4.f) - This service provides a means of assigning process agents to enact various processes and process elements, and this is typically done under constraints of time, budget, manpower assignments, equipment suites, and process definition technology (e.g., the formality or completeness of the installed process description language may be insufficiently unambiguous for totally automated enactment).

Communication Service (8.5) - This service provides a standard communication mechanism which may be used for inter-tool and inter-service communication. The services depend upon the form of communication mechanism provided: messages, process invocation and remote procedure call, or data sharing. (This service taken from the NIST/ECMA Framework Model [24])

User Interface Service (8.6) - The subject of user interfaces is an extremely complex issue which is far more general than integration frameworks. Nevertheless, a consistent User Interface Service may be adopted for a complete framework. (These services taken from the NIST/ECMA Framework Model [24])

User Interface Metadata Service (8.6.a) - This service provides for describing the objects used by the User Interface Services.

Session Service (8.6.b) - This service provides the functionality needed to initiate and monitor a session between the user and the environment.

Security Service (8.6.c) - This service provides the security constraints needed by the UI.

Profile Service (8.6.d) - This service provides the tool-to-session transformations needed to run multiple tools on multiple UI devices.

User Interface Name and Location Service (8.6.e) - This service permits the framework to manage multi-user and multi-platform environments. It permits various sessions to communicate with various tools and various display devices.

Application Interface Service (8.6.f) - This service provides most of the data transfer capabilities into and out of the tools and environment to the end user.

Dialog Service (8.6.g) - This service provides for integrity constraints between the user and the framework.

Presentation Service (8.6.h) - This service provides for low-level manipulation of display devices by the user interface.

Internationalization Service (8.6.i) - This service provides capabilities concerned with different national interests.

User Assistance Service (8.6.j) - This service provides a consistent feedback from various tools to the user for help and error reporting.

User Command Interface Services (8.7) - These services provide the functionality of a traditional shell. (These services taken from IEEE P1003.0 [10])

Network Services (8.8) - These services provide for the transfer of information among processes within a distributed environment. (These services taken from IEEE P1003.0 [10])

Directory Services (8.8.a) - This service allows for the names and addresses of objects to be accessed by an application.

Application to System Services (8.8.b) - This service provides support to an application but not directly controlled by it.

Application to Application Services (8.8.c) - This service supports the actual reading and writing of data across the network.

Data Representation Services (8.8.d) - This service provides data conversion.

Distributed System Services (8.8.e) - This service allows for the identification and use of resources in a distributed system.

Network Management Services (8.8.f) - This service manages the network objects and relationships.

Modem/dialup Services (8.8.g) - This service provides vendors' assistance to users.

Appendix B: Models of Tool Assessment and Evaluation

While no organization has published a model of SEE assessment, several groups have published models or criteria for software engineering tool assessment, and one (STSC) has published a model of SEE framework assessment. Two tool assessment models follow providing background information on recognized tool assessment methods since a full SEE contains tools and the closest models to work from in designing a SEE assessment method are those in tool assessment. The first model described is the model presented in the ISO and IEEE proposed standards of tool assessment. The second is the model employed by the Software Technology Support Center (STSC) for its annual assessment of software engineering tools. There are many similarities between the models, both emphasize tailoring the model and the criteria for assessing the SEE tools to meet the needs of the actual users of the tools and both emphasize a sharp division between evaluation and selection of tools in the assessment process.

B.1 IEEE and ISO Recommended Tool Evaluation and Selection Processes

IEEE has developed and ISO is developing very similar recommended practices for tool evaluation and selection. The tool evaluation and selection models presented in the IEEE and ISO documents emphasizes generality and flexibility. The IEEE/ISO model is intended for use in evaluating and selecting any type of CASE tool. To enhance the model's flexibility, there is a sharp division between the evaluation and selection processes allowing the application of the two processes separately or as a unified sequence. When an organization applies the two processes as a sequence, the evaluator performs the evaluation process and immediately passes the information to a selector who uses the selection process to select a tool for purchase. When the two processes do not have an immediate connection, the evaluation process generates data maintainable for future use and the selection process uses previously generated data.

B.1.1 Evaluation process

The purpose of the evaluation process is to assess candidate tools for performing a particular function with respect to selected criteria. Both the IEEE and ISO documents contain lists of suggested criteria including both objective and subjective criteria. The list is not meant to be exhaustive nor are all the criteria applicable to all the tools. The IEEE/ISO model assumes the evaluator will augment the criteria as necessary with criteria related to the functions the buying organization needs from the tool. Choosing good criteria and standards against which to measure the criteria requires that the evaluator exercise his judgement and expertise. The result of the evaluation process is a record of the evaluation of each tool with respect to each criteria as measured against a specified standard using a specified method. The steps of the evaluation process follow:

1) Prepare an evaluation task definition statement including
    the purpose of the evaluation
    the scope of the evaluation
    any assumptions and constraints
2) Identify the evaluation criteria
3) Identify the candidate CASE tools
4) Evaluate the candidate CASE tools
5) Report Results including
    tool information
    evaluation background
    evaluation approach
    evaluation steps
    specific results
    summary

The evaluation task definition statement defines the users needs and constraints to the evaluators. This is an extremely important document since the user requirements determine how the evaluator performs his job. As part of the process of selecting criteria to match the user's requirements and differentiate among the tools' capabilities, the evaluator must establish a test plan for applying the chosen criteria to the candidate tools and the possible set of resulting values for each criteria.

Once the evaluator has identified candidate CASE tools and has a test plan for applying the criteria, the evaluator applies the criteria to the candidate CASE tools and reports the results. How the evaluator applies the criteria depends on the user and evaluator's definitions of the requirements and the evaluator's test plan. For objective criteria, the evaluator should establish a repeatable procedure. If the criteria is subjective, more than one evaluator should apply the criteria to the CASE tool to attempt to get a consensus opinion. Once the evaluator obtains his results, he reports them in a formal manner to the user, and forwards the results as necessary to the selector.

B.1.2 Selection process

The purpose of the selection process is to identify the best CASE tool for the user's needs from among the evaluated candidate CASE tools. The selection process assumes that all the relevant candidate CASE tools have gone through the evaluation process and the results are available; however, the selection process does not assume that the evaluation was performed solely for the purpose of a making a selection at a particular time. The steps of the selection process follow:

1) Prepare a selection task definition statement including
    the purpose of the selection
    the scope of the selection
    any assumptions and constraints
2) Identify and weight selection criteria
3) Identify candidate CASE tools
4) Get evaluation results
5) Apply weighted selection criteria to evaluation results
6) Iteration
7) Recommend selection

Once again, the user must begin the process by deciding what are the needs, assumptions and constraints and their relative importance for the desired CASE tool. If the user has absolute constraints of the CASE tool (e.g., in terms of price or support for a particular software development methodology), the user must make this clear to the selector. The selector uses the selection task definition statement to identify and weight criteria reflecting the user's requirements.

The selector can use the existing evaluations of CASE tools to decide which CASE tools are potentially appropriate to the user's need, select their evaluation results and apply weights to the criteria to rate the candidate tools. However, both the IEEE and ISO documents emphasize two potential problems with using a preexisting evaluation. First, the evaluations could be out of date either because vendors have released new versions of tools or new tools have come on the market. Second, since the evaluations may not have been performed for this particular selection, the evaluations may not have applied the needed criteria to the tools.

Even if the evaluation process is reentered to rectify shortcomings, the selection process may not produce any tool meeting minimal user needs. The user may realize the evaluation or selection task definition statements do not accurately reflect the software development needs of the organization in acquiring the tool. Alternatively, the user may find that the criteria and weights on the criteria do not adequately reflect the requirements. As long as the user does not require the use of new criteria the selection process iterates until the user and selector agree on the weighted criteria for the process. When they agree on the weights and criteria, presumably, the user and selector will agree that the selection was the best available on the market at the time. If the user requires the use of criteria which were not part of the original evaluation criteria, the evaluator must restart the evaluation process.

Both the IEEE and ISO recommended practices for tool evaluation and selection emphasize the separation of the two processes, the equal importance and complexity of the two processes and the selection and weighting of criteria reflective of user needs and goals.

B.2 Software Technology Support Center (STSC) Model

STSC developed its Test and Evaluation (T&E) process for CASE tools with the long

term goal of improving software development for Air Force Systems by helping software developers make appropriate tool selections. The immediate purpose of publishing the T&E process and the STSC's tool evaluations obtained from this process are to give software professionals better information for making choices by increasing comparability, consistency and repeatability of software tool evaluations. In the longer term, STSC intends that its model make the tool evaluation process more efficient through evaluation reuse and that it facilitate feedback to software tool developers. The results of the process are the information used to perform the assessment and the formally reported results of applying the criteria and executing the test plan on the tools. To enhance the reusability of the evaluation, the STSC reports the results in pre-defined forms. The steps of the T&E process follow:

1) Analysis
    identify and compile a long list of available tools
    develop tool characteristics common to domain
    develop a set of essential characteristics
    compile short list of tools containing essential characteristics
2) Assessment
    identify all short list tool's functional characteristics through
        vendor surveys
        user interviews
        documentation reviews
        informal hands-on tests and demonstrations
    record characteristics in Tool Capability Matrix (TCM)
    obtain expert user critiques of short list tools
3) Evaluation Guidance
    design a test plan for evaluating tools
    develop tool-independent guidelines for evaluation each characteristic
    consolidate test plan and evaluation guidelines into Test and Evaluation Guideline (TEG)
4) Detailed Evaluation
    develop tool-specific Test and Evaluation Procedures (TEP) from TEG
    perform TEP
    update TCM
    record evaluations in Tool Evaluation Matrix (TEM)
    produce tool evaluation report (TER)
5) Recommendation
    accumulate user weights for characteristics
    check that TCM and TEM are current
    apply tool weights to tool characteristic scores
    annotate TER's with tool rank
    develop Tool Comparison Report (TCR) with
        tool information
        assessments
        critiques

evaluations

6) Selection

The first step of the T&E process is an analysis of both the user's needs in a specific CASE tool domain and the characteristics of the CASE tools in that domain. The T&E process differs from the IEEE and ISO processes by building up a list of criteria from examining available CASE tools for a particular set of tasks and determining what characteristics apply to or distinguish them. Once the T&E process establishes the important characteristics of CASE tools for specific tasks, the potential user, for the STSC the Air Force, can identify those criteria from the characteristics which are important for the user's needs. Only those tools possessing the available tool characteristics that the user deemed essential in the first phase need further consideration. This process emphasizes availability of the tools over absolute user requirements.

In order to provide more detailed evaluations of tools which met the essential characteristics, the T&E process requires the development of guidelines and a formal test plan in the third phase. These documents define specific methods for formally testing the tool characteristics. The evaluator executes the test plan according to the guidelines in the fourth phase. The STSC intends the test plan to be a standard method for evaluating tools in a specific CASE tool domain customized by guidelines indicating the user's important criteria.

Once the evaluator has executed the test plan and raw scores are available for the characteristics, the evaluator in consultation with the user performs the fifth and sixth phases of the process. The evaluator obtains information from the user about the relative importance of the tool criteria and applies weights to the criteria. These weights are in turn applied to the raw scores obtained from the evaluation. The evaluator ranks the tools and the user can make a selection from that ranking.

The STSC model differs from the IEEE and ISO models of tool assessment in that the results of the assessment are more than the tool evaluations and recommendations. The STSC process artifacts are defined characteristics of tools in the CASE tool domain, criteria reflecting user needs in that domain, default weights on those criteria, domain specific weights on the criteria, a test plan for evaluating tools in that domain, and guidelines for executing the test plan in that domain reflecting a particular buying organizations requirements. These process artifacts are available for reuse in later evaluations.

Appendix C: Mapping Guidelines

The approach to SEE assessment and evaluation used in this paper rely heavily on mapping the candidate SEEs and the user's requirements to the reference models. There is a set of mapping guidelines [16] and published examples of SEE mappings to an early edition of the NIST/ECMA Framework Model [e.g., 4]. There is a notation for describing mappings published in [16], but it is too cumbersome for analyzing the resulting documents.

From experience in performing mappings some specific guidelines follow:

1) Services and SEE capabilities, especially framework capabilities rarely correspond exactly. One service may correspond to more than one SEE capability or component, or one SEE capability or component may correspond to several services.

2) A SEE may only partially provide a service. It is important that the notation selected account for partial or incomplete services and define the extent of the provided service.

3) A SEE capability designed to work with another capability does not automatically supply that other capability. For example, if a data dictionary in a SEE provides a capability to represent metadata describing what is in an information repository, it provides the *metadata service*. The data dictionary does not automatically provide the *data storage service* just because metadata can be stored and retrieved.

4) If a SEE does not provide a *framework* service, the information should be highlighted. A missing framework service indicates a limitation on the extent of SEE integration possible or a limitation on the supported SEE functions.

5) If a reference model service and a SEE capability have the same name, they do not necessarily provide the same functionality and should not automatically be mapped. This may seem a trivial point, but it can easily be overlooked. Terminology in the SEE area is not well-defined or established.

6) Service dimensions are different views of the same service. Each view presents new information on the SEE capability but doesn't change the basic function of the capability. Information describing a capability can map to a service dimension, but the capability itself maps to a service. Each dimension in a provided service exists in a SEE product, but information for some of the dimensions may be unavailable.

In mapping SEE capabilities to reference model services, the assessor must note where the SEE capabilities and reference model services do not correspond well and where the assessor must exercise his judgement. Especially, if the assessor is working from documentation or specifications, the exact nature of the capability provided may be unclear. Whenever the assessor has to exercise judgement in establishing correspondence, the reasons for that judgement should

be noted.

The most important result of a mapping often is not necessarily an identification of the services provided, but identification of the services that are missing. Especially for framework services, missing services mean possible limitations on the capacity of the SEE especially its capacity to integrate tools. It may not be possible to expand the framework later with new groups of capabilities. If a framework has been in use for some time and the vendor adds new capabilities, the tools relying on the old framework may not be compatible. Even if adding the capabilities is possible, the tool vendors will not have designed tools for that framework with the capability in place and will not expect to use it. Missing tools are less important since, especially with open systems, tools can be added to a framework later as needed.

Appendix D: Criteria Sources

The following are sources for criteria. These criteria sources are organized by the applicable reference model services. The documents often overlap several services but predominately relate to one service or service group. A category for general criteria is added to include criteria which do not refer to capabilities found in the reference models.

D.1 Technical Engineering Services Criteria Sources:

All Services:

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993.

System Engineering Services (4.1):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993.

System Requirements Engineering Service (4.1.1):

Requirements Analysis and Design Tools Report. STSC: April, 1992.

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

System Design and Allocation Service (4.1.2):

Requirements Analysis and Design Tools Report. STSC: April, 1992.

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

System Simulation and Modeling Service (4.1.3):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

System Static Analysis Service (4.1.4):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.2)

Source Code Static Analysis Tools Report. STSC: April, 1992.

System Testing Service (4.1.5):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.3)

System Re-Engineering Service (4.1.7):

Re-Engineering Tools Report. STSC: April, 1992.

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

Traceability Service (4.1.10):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

Software Engineering Services (4.2):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993.

Software Requirements Engineering Service (4.2.1):

Requirements Analysis and Design Tools Report. STSC: April, 1992.

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

Software Design Service (4.2.2):

Requirements Analysis and Design Tools Report. STSC: April, 1992.

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

Software Simulation and Modeling Service (4.2.3):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

Software Generation Service (4.2.5):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.2)

Software Compilation Service (4.2.6):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.2)

Software Static Analysis Service (4.2.7):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

Source Code Static Analysis Tools Report. STSC: April, 1992.

Software Debugging Service (4.2.8):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.2)

Software Testing Service (4.2.9):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.3)

Software Build Service (4.2.10):

Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems, Inc., 1993.

Software Reverse Engineering Service (4.2.11):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.2)

Re-Engineering Tools Report. STSC: April, 1992.

Software Re-Engineering Service (4.2.12):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.2)

Re-Engineering Tools Report. STSC: April, 1992.

Software Traceability Service (4.2.13):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.2.1)

Life-Cycle Process Engineering Services (4.3):

Smith, Dennis, et al. Software Engineering Environments Evaluation Issues. SEI Technical Report, 1993.

Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems, Inc., 1993.

D.2 Technical Management Services Criteria Sources

Configuration Management Service (5.1):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.4.2)

Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems, Inc., 1993.

Change Management Service (5.2):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.4.2)

Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems, Inc., 1993.

Reuse Management Service (5.3):

Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems, Inc., 1993.

D.3 Project Management Services Criteria Sources

All Services:

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.1)

Project Management Tools Report. STSC: March, 1992.

D.4 Support Services Criteria Sources

Common Support Services (7.1):

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.4.1)

Documentation Tools Report. STSC: March, 1992.

Publishing Service (7.2):

Documentation Tools Report. STSC: March, 1992.

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.1.4.1)

D.5 Framework Services Criteria Sources

All Services:

Software Engineering Environments Report. STSC: March, 1992.

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993. (Section 9.3)

Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems, Inc., 1993.

Smith, Dennis, et al. Software Engineering Environments Evaluation Issues. SEI Technical Report, 1993.

D.6 General Criteria Sources

Evaluation and Selection of CASE Tools. Draft ISO Recommended Practice, Project 7.25, 1993.

Evaluation Criteria for COTS Software Engineering Frameworks. Draft Report. International Software Systems, Inc., 1993.

Smith, Dennis, et al. Software Engineering Environments Evaluation Issues. SEI Technical Report, 1993.

D.7 Criteria Sources for Management Readiness to Adopt SEE Technology

Smith, Dennis, et al. Software Engineering Environments Evaluation Issues. SEI Technical Report, 1993.